

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号  
特開2000-228768  
(P2000-228768A)

(43) 公開日 平成12年8月15日 (2000.8.15)

(51) Int.Cl.<sup>7</sup>

H 0 4 N 7/24

識別記号

F I

H 0 4 N 7/13

テーマコード(参考)

Z 5 C 0 5 9

審査請求 未請求 請求項の数 5 O L (全 27 頁)

(21) 出願番号 特願平11-29377

(22) 出願日 平成11年2月5日 (1999.2.5)

(71) 出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72) 発明者 田原 勝己

東京都品川区北品川6丁目7番35号 ソニ

ー株式会社内

(72) 発明者 村上 芳弘

東京都品川区北品川6丁目7番35号 ソニ

ー株式会社内

(74) 代理人 100082131

弁理士 稲本 義雄

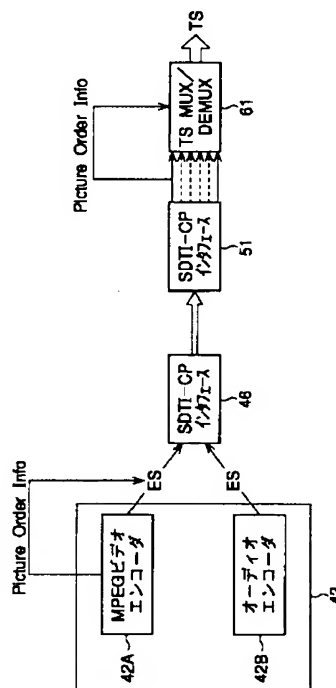
最終頁に続く

(54) 【発明の名称】 デジタル信号伝送装置および方法、並びに提供媒体

(57) 【要約】

【課題】 所定のスタジオと他のスタジオの両方において、画像データを迅速に処理できるようにする。

【解決手段】 MPEGビデオエンコーダ42Aは、エンコードしたビデオデータのエレメンタリストリームに、画像の符号化順序と表示順序を表すPicture Order Informationを重畳し、SDTI-CPインタフェース46を介して、同一スタジオ内のSDTI-CPネットワークに伝送する。TS MUX/DEMUX 61は、SDTI-CPインタフェース46、51を介して、エレメンタリストリームの供給を受けたとき、そこに含まれているPicture Order Informationを抽出し、それに基づいてタイムスタンプを生成し、トランスポートストリームに重畳して、他のスタジオに伝送する。



## 【特許請求の範囲】

【請求項1】 画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する第1の受信手段と、

前記第1の受信手段により受信された前記第1のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する第1の出力手段とを含むことを特徴とするデジタル信号伝送装置。

【請求項2】 前記第2のビットストリームを受信する第2の受信手段と、

前記第2の受信手段により受信された前記第2のビットストリームから、前記タイムスタンプを生成し、前記第1のビットストリームに挿入して出力する第2の出力手段とをさらに含むことを特徴とする請求項1に記載のデジタル信号伝送装置。

【請求項3】 前記符号化順序は、前記画像信号のフィールドの表示時間を単位とすることを特徴とする請求項1に記載のデジタル信号伝送装置。

【請求項4】 画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する受信ステップと、前記受信ステップの処理で受信された前記第1のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する出力ステップとを含むことを特徴とするデジタル信号伝送方法。

【請求項5】 画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する受信ステップと、前記受信ステップの処理で受信された前記第1のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する出力ステップとを含む処理をデジタル信号伝送装置に実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする提供媒体。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、デジタル信号伝送装置および方法、並びに提供媒体に関し、特に、異なるスタジオ間で画像を送受信し、処理する場合において、送受信する画像信号のシステム全体における遅延量をできるだけ少なくするようにした、デジタル信号伝送装置および方法、並びに提供媒体に関する。

## 【0002】

【従来の技術】画像信号または音声信号を符号化して伝

送する場合、ISO/IEC11172(MPEG-1)もしくはISO/IEC13818(MPEG-2)に示されている符号化方式が用いられることが多い。画像信号を、MPEG(Moving Picture Experts Group)方式で符号化する場合の技術として、コーディングフェーズ(coding phase)と両方向予測があげられる。

【0003】coding phaseとは、図1に示すように、1枚の画像の画素エリア1のうち、符号化が行われる範囲としての有効画素エリア2を規定するためのコードであり、具体的には、全てのラインのうち、有効画素エリア2の最初のライン(図1のV-phase[line]の矢印で示されたライン)、および、全てのサンプル(画素)のうち、有効画素エリア2の最初のラインの最初のサンプル(図1のH-phase[sample]の矢印で示されたサンプル(画素))を示すものである。画素エリア1の最後のラインを、V-Phase[Lmax]とし、画素エリア1のラインの最後のサンプル(画素)を、H-Phase[Smax]とすると、符号化は、例えば、垂直方向に、V-Phase[line]から、(V-Phase[Lmax] - V-Phase[line]+1)までの範囲で行われ、水平方向に、H-Phase[sample]から、(H-Phase[Smax]-H-Phase[sample]+1)までの範囲で行われる。

【0004】また、一般の例えば、NTSC方式のテレビジョン受像機などの画像信号において、図1の画像信号のフレームは、図2に示すように、2枚のフィールド(図2のフィールド1およびフィールド2)から構成されている。フィールド1には、例えば、奇数ラインのデータが表示され、フィールド2には偶数ラインのデータが表示される。2枚のフィールドは、それぞれ、補助データ(ancillary data)と、画像(イメージ)データ(image data)から構成されている。

【0005】補助データは、耳の不自由な人のための文字放送用のテレテキストデータ、タイムコード、または映画などの字幕のクローズドキャプションデータに利用され、ブランキング区間(例えば、各フィールドの第10ライン乃至第22ライン)(図2においては、いずれのフィールドも、そのフィールド内の上から順番の番号でラインが表されている)のうちの所定のラインに挿入されている。一般に、各フィールドの第23ラインより図中下に位置するラインの部分(実際に、画像として表示される部分)は、画像データとして、MPEG方式などにより符号化される。

【0006】前述したMPEG符号化方式は、画像データにのみ適用され、coding phase、および補助データについては、MPEGの規格には、特に明確に記述(規定)されていない。そのため、coding phaseは、自由度を有し、様々なアプリケーションによって異なっている。

【0007】このような画像信号を、符号化または復号するときのシステム構成を、図3を参照して説明する。アプリケーションAの画像データは、アプリケーション

10

20

30

40

50

A用MPEGエンコーダ11で符号化され、エレメンタリストリーム(ES)として出力され、アプリケーションA用MPEGデコーダ12で復号される。アプリケーションBの画像データは、アプリケーションB用MPEGエンコーダ13で符号化され、エレメンタリストリーム(ES)として出力され、アプリケーションB用MPEGデコーダ14で復号される。

【0008】つまり、あるアプリケーションの画像データは、そのアプリケーション専用のエンコーダ(例えば、アプリケーションA用MPEGエンコーダ11)で符号化され、エレメンタリストリーム(ES)として、そのアプリケーションに対応した専用のデコーダ(例えば、アプリケーションA用MPEGデコーダ12)に出力される。専用デコーダに入力されたエレメンタリストリーム(ES)は、そのデコーダが有している、そのアプリケーションのcoding phaseに関する情報に基づき、復号される。

【0009】次に、現在主流となっている補助データの伝送方式について、図4を参照して説明する。画像信号は、MPEGエンコーダ21に入力され、分離部21において、画像データと補助データに分離される。MPEGエンコード部23は、画像データをMPEG符号化する。補助データは、可変長符号化部24で、MPEGエンコード部23より出力されたMPEG方式のトランスポートストリーム中のuser dataに挿入される。このトランスポートストリーム中のuser dataは、符号化された画像データ(ピクチャ)単位に挿入(記述)できるため、補助データは、対応する各符号化画像データのフレームのuser dataごとに挿入される。

【0010】補助データが挿入されたuser dataを含むトランスポートストリームは、所定の伝送路を介して伝送され、MPEGデコーダ25に入力される。MPEGデコーダ25内の可変長復号部26は、補助データと画像データとを分離する。画像データはMPEGデコード部27で復号され、合成部28で補助データと合成されて、画像信号として、図示せぬ表示装置に出力される。

【0011】次に、MPEG方式の予測について説明する。両方向予測に基づいて生成された符号化画像データのピクチャはBピクチャと称される。Bピクチャは、時間的に前または後に位置する2枚の参照画像データから予測されて生成される。前方予測に基づいて生成された符号化画像データのピクチャはPピクチャと称される。Pピクチャは、時間的に前に位置する1枚の参照画像データから予測されて生成される。予測が行われず、そのまま画像データが符号化(イントラ符号化)された符号化画像データのピクチャはIピクチャと称される。つまり、入力された画像データは、Bピクチャ、Pピクチャ、またはIピクチャのいずれかの符号化画像データに符号化される。

【0012】両方向予測および前方予測について、図5

を参照して説明する。図5の例では、1つのGOP(Group of Picture)が、9枚のピクチャから構成されている。図5の上段は、符号化画像データが生成されるときの、予測の方向(依存関係)を示す予測構造を表しており、図5の下段は、実際に画像データが符号化される順序を示す符号化構造を表している。Bピクチャ、またはPピクチャは、時間的に前または後の画像データから予測されて生成されるために、Bピクチャ、またはPピクチャだけでは、符号化を行うことはできない。すなわち、Bピクチャ、またはPピクチャは、参照画像データとの差分をデータとする符号化画像データであるので、Bピクチャ、またはPピクチャだけでは、画像データを復号することもできない。

【0013】予測の依存関係について詳細に説明すると、例えば、GOP(N-1)において、表示順序が先頭のBピクチャは、先頭から3番目のIピクチャと、図示されていないGOP(N-2)(GOP(N-1)の直前のGOP)の最後のPピクチャから予測されて符号化される。先頭から2番目のBピクチャも同様に、先頭から3番目のIピクチャと、図示されていないGOP(N-2)の最後のPピクチャから予測されて符号化される。先頭から3番目のIピクチャは、そのまま符号化(イントラ符号化)される。先頭から4番目、および5番目のBピクチャは、先頭から3番目のIピクチャと先頭から6番目のPピクチャから予測されて符号化される。先頭から6番目のPピクチャは、先頭から3番目のIピクチャから予測されて符号化される。

【0014】つまり、予測構造において、Bピクチャ(例えば、先頭のBピクチャと先頭から2番目のBピクチャ)が符号化されるには、予測の参照画像データ(例えば、3番目のIピクチャとGOP(N-2)の最後のPピクチャ)が、先に符号化されている必要がある。すなわち、図5の下段に示すような、符号化構造の順番で符号化されなければならない。そのため、MPEGエンコーダは、符号化(エンコード)時、IピクチャとPピクチャの間に存在する、2枚の連続したBピクチャを符号化するために必要な参照画像データ(先頭から3番目のIピクチャ)が入力されるまで、2枚のBピクチャをバッファリングする(Iピクチャが入力されるまで、符号化の開始を遅延させる)必要がある。このバッファリングにより、MPEGエンコーダにおいて、入力された画像信号を符号化するとき、参照画像データに挟まれたBピクチャの枚数(=2)+1枚(合計3枚)の遅延時間が発生する。

【0015】このMPEGエンコーダにおいて発生する遅延について、図6を参照して、さらに詳細に説明する。図6の上段はMPEGエンコーダ31における、入力画像(画像データ)の入力順番(表示順序)とその種別を表しており、図6の中段は、入力画像(画像データ)が符号化された符号化画像データの順番を表している。

【0016】MPEGエンコーダ31は、時刻t4のPピク

チャが入力されるまで（Bピクチャの枚数（＝2）＋1枚分の時間だけ）、時刻t1に入力されたIピクチャの符号化を遅延させる（前述のバッファリングを行う）。すなわち、MPEGエンコーダ31は、時刻t4で時刻t1に入力されたIピクチャを符号化し、時刻t5で時刻t4に入力されたPピクチャを符号化し、時刻t6で時刻t2に入力されたBピクチャを符号化し、時刻t7で時刻t3に入力されたBピクチャを符号化する。その後、MPEGエンコーダ31は、入力画像を図6の中段の符号化画像データの順番で、順次、符号化する。

【0017】このように、MPEGエンコーダ31は、入力画像に対して、予測に必要な参照画像データから順次符号化する。つまり、MPEGエンコーダ31において、入力画像（画像データ）は、入力画像の順番から、符号化される画像の順番に並べ換えられ、図6の中段に示すように、入力画像が符号化された順番にビットストリームとして、MPEGデコーダ32に出力される。

【0018】このように、入力画像が符号化される順番と表示される順番は一致しないので、MPEGにおいては、符号化順序を表すDTS（Decoding Time Stamp）と、表示順序を表すPTS（Presentation Time Stamp）が、トランスポートストリーム中に挿入されるようになされている。入力画像の符号化順序と表示順序の関係をさらに説明すると、いま、符号化順序をフレーム単位で表すものとする、図6に示すように、入力画像は符号化される順番で付番され、時刻t4で符号化されたIピクチャの符号化順序の値は“1”となり、時刻t5で符号化されたPピクチャの符号化順序の値は“2”となり、時刻t6で符号化されたBピクチャの符号化順序の値は“3”となる。以下、同様に符号化される順番に符号化順序が付番される。DTSはフレーム単位で符号化順序を表しているわけではないが、ほぼ、この符号化順序に対応する。DTSはまた、ビットストリームを音声信号などと多重化して出力するとき、MPEGデコーダ32において、復号する順番として使用される。

【0019】表示順序は、画像データが復号されて表示される順番（入力画像データの順番と同じ）となる。具体的には、時刻t4で符号化されるIピクチャ（符号化順序＝1）の表示順序は、時刻t5でPピクチャ（符号化順序＝2）が符号化されるときに表示されなければならないので、そのPピクチャの符号化順序と同じ値“2”となる。時刻t5で符号化されるPピクチャ（符号化順序＝2）の表示順序は、時刻t8でPピクチャ（符号化順序＝5）が符号化されるときに表示されなければならないので、そのPピクチャの符号化順序と同じ値“5”となる。時刻t6で符号化されるBピクチャ（符号化順序＝3）は、符号化されると、直ちに復号されて表示されなければならないので、符号化順序と表示順序は同一となり、表示順序の値は“3”となる。時刻t7で符号化されるBピクチャ（符号化順序＝4）も同様に、

表示順序＝符号化順序＝4となる。PTSはフレーム単位で表示順序を表しているわけではないが、ほぼ、この表示順序に対応する。PTSはまた、MPEGデコーダ32において、復号後、出力（表示）する順番として使用される。

【0020】MPEGデコーダ32では、符号化順序が連続する2枚のIピクチャもしくはPピクチャのうち、最初の1枚目は、2枚目が復号されるとき、表示される。例えば、MPEGエンコーダ31が出力したビットストリーム中の符号化画像データの1枚目のIピクチャ（符号化順序＝1、表示順序＝2）と、2枚目のPピクチャ（符号化順序＝2、表示順序＝5）は、符号化順序が連続しており、2枚目のPピクチャが符号化されるとき（時刻t5のとき）、1枚目のIピクチャが復号されて表示される。

【0021】このように、MPEGエンコーダ31は、入力画像（画像データ）の予測構造（参照画像データに挟まれているBピクチャの枚数）を知っているので、画像データに符号化順序（DTS）と表示順序（PTS）を付番することができる。

【0022】放送局などのスタジオ内で、図7に示すシステム構成で、MPEG符号化されたビットストリームを送送することが考えられている。スタジオ41内のMPEGエンコーダ42、MPEGエンコーダ43、MPEGデコーダ44、およびMPEGデコーダ45は、それぞれ、SDTI-CP（Serial Data Transfer Interface - Content Package）インタフェース46乃至49を介して、SDTI-CPネットワーク（例えば、同軸ケーブルにより構成されるネットワーク）50に接続される。SDTI-CPネットワーク50は、SDI（Serial Data Interface）をベースとした270Mbpsの伝送速度を有し、MPEG方式のエレメンタリストリーム（ES）をそのまま伝送することが可能であり、スタジオ内のような閉じたネットワークに適している。

【0023】スタジオ41において、例えば、MPEGエンコーダ42は、MPEG符号化されたエレメンタリストリーム（ES）を、SDTI-CPネットワーク50を介して、MPEGデコーダ44、MPEGデコーダ45に伝送することができる。

【0024】このSDTI-CPネットワーク50で伝送されるエレメンタリストリーム（ES）は、図8に示す構造となっており、画像信号のフレーム単位で、画像データ（図8において薄い影を付けた部分）と音声データ（図8において濃い影を付けた部分）がバッキングされており、フレームシンク（図8の点線）により区切られたフレーム境界で、簡単に編集を行うことができる。このエレメンタリストリーム（ES）における画像データと音声データは、イントラ（フレーム内符号化）処理されたデータである。

【0025】図7に示すスタジオ41のシステムは、そのMPEGエンコーダ42、43とMPEGデコーダ44、45

が、SDTI-CPインタフェース46乃至49を介して、SDTI-CPネットワーク50に接続されており、図3に示したような、アプリケーションごとに専用のエンコーダ（例えば、アプリケーションA用MPEGエンコーダ11）とデコーダ（例えば、アプリケーションA用MPEGデコーダ12）が1対1に対応するシステムとは構成が異なっている。すなわち、デコーダ（例えば、図7のMPEGデコーダ44）は、エンコーダ（例えば、図7のMPEGエンコーダ42またはMPEGエンコーダ43）で、様々なアプリケーションの画像信号が符号化された、エレメンタリストリーム（ES）を受け取ることができる。

【0026】図7のシステム構成を、図3のシステム構成に対応させた図9を参照して説明すると、アプリケーションA用MPEGエンコーダ42で符号化されたエレメンタリストリーム（ES）と、アプリケーションB用MPEGエンコーダ43で符号化されたエレメンタリストリーム（ES）は、それぞれSDTI-CPインタフェース46、47に入力される。SDTI-CPインタフェース46、47は、それぞれMPEG符号化されたエレメンタリストリーム（ES）を、SDTIフォーマットのエレメンタリストリーム（ES）に変換して、SDTI-CPネットワーク50を介して伝送する。SDTI-CPインタフェース48は、SDTIフォーマットのエレメンタリストリーム（ES）を、MPEG符号化されたエレメンタリストリーム（ES）に変換し、MPEGデコーダ44に出力する。

【0027】MPEGデコーダ44は、それぞれ入力されたアプリケーションA用のエレメンタリストリーム（ES）とアプリケーションB用のエレメンタリストリーム（ES）を復号する。

【0028】ところで、補助データがトランスポートストリーム中のuser dataに挿入される場合、補助データは、対応する符号化画像のフレーム単位で挿入されるため、その挿入は1フレーム（2フィールド）ごととなる。

【0029】符号化される信号が、3-2プルダウン処理（例えば、24Hzのフレームレートを持つ映画の画像信号を、30Hzのフレームレートを持つNTSC方式の画像信号に変換する処理）された信号である場合、その信号は、図10に示すように、24Hzの各フレームを、交互に、リピートフィールドが作成されていない2フィールドのフレーム、またはリピートフィールドが作成されている3フィールドのフレームとすることで、30Hzの信号とされている。

【0030】例えば、図4のMPEGエンコーダ21は、3-2プルダウン処理により30Hzのフレームレートに変換された画像信号が入力されたとき、フィールドの繰り返しを検出して、元の24Hzの符号化フレーム単位で符号化を行い、その処理に対応して、フラグ（Repeat\_first\_field, Top\_field\_first）を生成する。

【0031】Repeat\_first\_fieldのフラグの“1”は、

リピートフィールドが作成されたことを意味し、Repeat\_first\_fieldのフラグの“0”は、リピートフィールドが作成されていないことを意味する。Top\_field\_firstのフラグは、フレームを構成するフィールドのうち、最初のフィールドがトップフィールドであるのか、またはボトムフィールドであるのかを表している。Top\_field\_firstのフラグの“1”は、トップフィールドがボトムフィールドより時間的に早いフレーム構造であることを表しており、Top\_field\_firstのフラグの“0”は、ボトムフィールドがトップフィールドより時間的に早いフレーム構造であることを表している。

【0032】図10に示すように、原信号のフレームが3フィールドのフレームである場合、対応する1つの符号化フレームには、同一位相の2枚のフィールド（3-2プルダウン処理によりコピーにより生成されたフィールドとそのコピー元のフィールド）が存在する。しかしながら、この元の1つのフレームを構成する3フィールドが符号化フレーム単位で符号化されるとき、各符号化フレームごとの補助データは、符号化フレーム単位で、1つの補助データとして、user dataに記述されるため、原信号の同一位相のフィールドに異なった補助データが記述されていても、その異なった補助データを区別することができなくなってしまう。

【0033】

【発明が解決しようとする課題】例えば、所定のスタジオで、MPEG方式でエンコードした画像を処理し、他のスタジオにトランスポートストリームとして伝送し、他のスタジオでも処理するような場合、画像データの伝送は、符号化された順序で行う必要があるが、トランスポートストリームとして画像データを伝送する装置は、通常、符号化順序を知らないので、入力された画像データを一旦バッファリングし、符号化順序を知る必要がある。その結果、バッファリングのために画像データの供給を受けてから、それを伝送するまでに時間がかかり、迅速な処理、特に、リアルタイム処理が困難となる課題があった。

【0034】本発明はこのような状況に鑑みてなされたものであり、画像データを迅速に処理することができるようにするものである。

【0035】

【課題を解決するための手段】請求項1に記載のデジタル信号伝送装置は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する第1の受信手段と、第1の受信手段により受信された第1のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する第1の出力手段とを含むことを特徴とする。

【0036】請求項4に記載のデジタル信号伝送方法

は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する受信ステップと、受信ステップの処理で受信された第1のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する出力ステップとを含むことを特徴とする。

【0037】請求項5に記載の提供媒体は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第1のビットストリームを受信する受信ステップと、受信ステップの処理で受信された第1のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入して出力する出力ステップとを含む処理をデジタル信号伝送装置に実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする。

【0038】請求項1に記載のデジタル信号伝送装置、請求項4に記載のデジタル信号伝送方法、および請求項5に記載の提供媒体においては、受信された第1のビットストリームから順序情報が抽出され、抽出した順序情報に基づいてタイムスタンプが生成され、第2のビットストリームに挿入される。

【0039】

【発明の実施の形態】図11は、相互に離れた場所に位置するスタジオ41とスタジオ71の間で、MPEG符号化されたビットストリームを伝送するシステムの例を表しており、図7における場合と対応する部分には、同一の符号を付してある。この例の場合、各スタジオ内のネットワークを、多重化装置（以下、TS MUX/DEMUXと称する）を介して、衛星やATM（Asynchronous Transfer Mode）などの公衆網と接続することで、ビットストリームの伝送を行うことができる。スタジオ71のMPEGエンコーダ72乃至SDTI-CPインタフェース79は、図7のスタジオ41のMPEGエンコーダ42乃至SDTI-CPインタフェース49に対応するものである。ここでは、その説明を省略する。

【0040】SDTI-CPネットワーク50におけるエレメンタリストリーム（ES）は、図12に示すように、TS MUX/DEMUX61により、188バイト単位のトランスポートストリーム（TS）に変換されて所定の伝送媒体を介して伝送され、伝送されたトランスポートストリーム（TS）は、TS MUX/DEMUX62で、SDTIフォーマットのエレメンタリストリーム（ES）に変換される。なお、図12において、薄い影を付けた部分は画像データのバケットを示しており、濃い影を付けた部分は音声データのバケットを示しており、影をつけていない部分は空きデータのバケットを示している。

【0041】図11において、スタジオ41内のMPEGエ

ンコーダ42の出力するエレメンタリストリーム（ES）が、TS MUX/DEMUX61でトランスポートストリーム（TS）に変換されるまでを、図13を参照して説明する。MPEGエンコーダ42内のMPEGビデオエンコーダ42Aは、MPEG符号化された画像データのエレメンタリストリーム（ES）を、SDTI-CPインタフェース46に出力し、オーディオエンコーダ42Bは、音声データのエレメンタリストリーム（ES）を、SDTI-CPインタフェース46に出力する。SDTI-CPインタフェース46は、入力されたエレメンタリストリーム（ES）をSDTIベースのフォーマットのエレメンタリストリーム（ES）に変換し、SDTI-CPネットワーク50を介して、SDTI-CPインタフェース51に出力する。SDTI-CPインタフェース51は、SDTIフォーマットのESをMPEG符号化のエレメンタリストリーム（ES）に変換し、TS MUX/DEMUX61に出力する。TS MUX/DEMUX61は、MPEG符号化のエレメンタリストリーム（ES）を188バイト単位のトランスポートストリーム（TS）に変換して伝送媒体に出力する。

【0042】図11において、TS MUX/DEMUX61から伝送されたトランスポートストリーム（TS）は、ATMなどの公衆網を介して、TS MUX/DEMUX62に投入され、MPEG符号化のエレメンタリストリーム（ES）に変換される。SDTI-CPインタフェース81において、MPEG符号化されたエレメンタリストリーム（ES）は、SDTIフォーマットのエレメンタリストリーム（ES）に変換され、スタジオ71のSDTI-CPネットワーク80に出力される。MPEGデコーダ74は、SDTI-CPインタフェース78を介して、スタジオ21から伝送されたエレメンタリストリーム（ES）を受信することができる。

【0043】図11のシステムにおいて、スタジオ41とスタジオ71の間で、画像信号を伝送する場合、多重化装置（例えば、図11のTS MUX/DEMUX61）には、画像データが符号化されたエレメンタリストリーム（ES）のみが入力されるので、多重化装置は、エンコーダ（例えば、図11のMPEGエンコーダ42）において行われた符号化順序の並べ変えの情報を知らない。このため、多重化装置は、入力されたエレメンタリストリーム（ES）を解釈して、エレメンタリストリーム（ES）をトランスポートストリーム（TS）に変換する処理をしなければならない。

【0044】このエレメンタリストリームを解釈する処理は、画像信号の符号化構造を把握する処理である。つまり、前述したエンコーダで入力画像をバッファリングして符号化し、ビットストリームに変換するときと同様の処理が、多重化装置でも必要となる。この多重化装置におけるバッファリング処理により、後段での画像データの復号までに遅延が発生し、リアルタイム処理が困難になる。

【0045】この多重化装置で発生する、システム構成上、問題となる遅延について、図14を用いて説明す

る。多重化装置に、図14に示す符号化順序で、エンコーダで符号化されたビットストリームが入力された場合、多重化装置は、2枚の連続したIピクチャ（時刻t4）とPピクチャ（時刻t5）が入力された後、さらにそれらに挟まれた2枚のBピクチャ（時刻t6と時刻t7）に続いてPピクチャ（時刻t8）が入力されるまで、表示順序を確定することができない。つまり、多重化装置は、図5に示したような符号化構造（IピクチャとPピクチャの間に2枚のBピクチャが挟まれている構造）を知らないので、2枚のBピクチャが入力され、その次のPピクチャ（時刻t8）が入力されたとき（Bピクチャの入力が終了したことを確認したとき）、時刻t4に入力されたIピクチャの表示順序（PTS）を確定することができる。

【0046】すなわち、この多重化装置においては、時刻t8になって初めて、符号化順序=1のIピクチャの表示順序=2を確定することができる。このため、多重化装置では、2枚の連続したIピクチャまたはPピクチャ+2枚のBピクチャの出現周期（=4）の遅延が発生する。この遅延は、エンコーダでの遅延とは別に新たに発生するため、システム全体として、エンコーダの遅延（=3）+多重化装置の遅延（=4）=7の大幅な遅延が生じる。

【0047】そこで本発明においては、符号化順序と表示順序を含む順序情報を、PictureOrder Infoとしてエレメンタリストリームに重畳するようにしている。この点のについては、後に詳述する。

【0048】図15は、本発明を適用したMPEGエンコーダ42の構成例を表している。MPEGエンコーダ42の分離部101は、入力された画像信号から、画像データと補助データを分離し、画像データをMPEGエンコード部103に出力し、補助データをエンコードコントローラ102に出力する。MPEGエンコード部103は、入力された画像データをMPEG方式により符号化するとともに、符号化順序を示すDTS\_counter、および表示順序を示すPTS\_counterを含むPOI（Picture Order Information）をエンコードコントローラ102に出力する。また、MPEGエンコード部103は、符号化した範囲の左上の位置（図1の有効画素エリア2の左上の画素の位置）を表すコーディングフェーズ（Coding Phase（V-Phase, H-Phase））をエンコードコントローラ102に供給する。

【0049】エンコードコントローラ102は、分離部101より供給された補助データに、それが属するフィールドを識別するField IDと、それが挿入されているラインを表すLine\_numberを付加し、その補助データと、MPEGエンコード部103より供給されたPOI、およびCoding Phase Information（CPI）を適宜処理し、user dataのフォーマットのデータとして、可変長符号化部104に出力し、多重化させる。

【0050】可変長符号化部102は、MPEGエンコード

部103より供給されたエンコードされた画像データを可変長符号化するとともに、エンコードコントローラ102より供給されたuser dataを画像データのエレメンタリストリームに挿入する。可変長符号化部104より出力されたエレメンタリストリームは、送信バッファ105を介して出力される。

【0051】図16は、本発明を適用したMPEGデコーダ44の構成例を表している。受信バッファ111は、入力されたデータを一旦バッファリングした後、可変長復号部112に出力する。可変長復号部112は、入力されたデータから、画像データとuser dataとを分離し、画像データをMPEGデコード部114に出力し、user dataをデコードコントローラ113に出力する。デコードコントローラ113は、user dataからPOIとCPIを分離し、それらをMPEGデコード部114に出力する。また、デコードコントローラ113は、user dataから分離した補助データを合成部115に出力する。MPEGデコード部114は、可変長復号部112より入力された画像データを、デコードコントローラ113より入力されたPOIとCPIを参照してデコードし、デコードした結果を合成部115に出力している。合成部115は、デコードコントローラ113より供給された補助データをMPEGデコード部114より供給された画像データと合成し、出力する。

【0052】以下に、MPEGエンコーダ42とMPEGデコーダ44の動作について説明する。

【0053】CPIをuser dataに記述して、エレメンタリストリーム（ES）に重畳して出力する動作を、図17を参照して説明すると、アプリケーションA用MPEGエンコーダ42は、アプリケーションAの画像信号を符号化したエレメンタリストリーム（ES）に、CPI（画像信号のフォーマットにおける有効画素エリアを示すデータであるV-PhaseとH-Phase）をuser dataに記述して（図15の可変長符号化部104がエンコードコントローラ102から出力されるCPIのデータをuser dataに記述して）、SDTI-CPインタフェース46に出力する。アプリケーションB用MPEGエンコーダ43も、MPEGエンコーダ42と同様に構成されており、アプリケーションBの画像信号を符号化したエレメンタリストリーム（ES）に、CPIをuser dataとして記述して、SDTI-CPインタフェース47に出力する。

【0054】SDTI-CPインタフェース46の構成例について、図18を参照して説明する（他のSDTI-CPインタフェースも同様に構成されている）。復号部121は、MPEGエンコーダ42から入力されたMPEG符号化されたエレメンタリストリーム（ES）を、符号化パラメータと画像データに分離し、画像データを復号し、符号化パラメータとともに符号化パラメータ多重化部122に出力する。符号化パラメータ多重化部122は、画像信号と符号化パラメータから、SDTI-CPベースのエレメンタリス



トリーム (ES) を生成して出力する。

【0055】SDTI-CPインタフェース46に、SDTI-CPベースのエレメンタリストリーム (ES) が入力された場合、符号化パラメータ分離部123は、エレメンタリストリーム (ES) から画像データと符号化パラメータを分離して、それぞれ符号化部124に出力する。符号化部124は、符号化パラメータを用いて、画像データを符号化し、MPEG符号化されたエレメンタリストリーム (ES) として出力するか、またはパケット化部125でMPEG符号化されたトランスポートストリーム (TS) として

出力する。  
【0056】SDTI-CPインタフェース46、47は、MPEG符号化されたエレメンタリストリーム (ES) を、SDTIフォーマットのエレメンタリストリーム (ES) に変換して、SDTI-CPネットワーク50を介して、伝送する。SDTI-CPインタフェース48は、SDTIフォーマットのエレメンタリストリーム (ES) をMPEG符号化されたエレメンタリストリーム (ES) に変換し、MPEGデコーダ44に出力する。

【0057】MPEGデコーダ44は、入力されたアプリケーションAのエレメンタリストリーム (ES) またはアプリケーションBのエレメンタリストリーム (ES) を復号し、それぞれのエレメンタリストリーム (ES) に記述されている(図16のデコードコントローラ113が出力する) CPIに基づいて、画像信号を有効画素エリアに配置するように復号する。

【0058】MPEGエンコーダ42、43は、それぞれが符号化したアプリケーションの画像信号のCPIをMPEG符号化されたエレメンタリストリーム (ES) 中のuser dataに記述することにより、画像データとともに、CPIを伝送することができる。また、MPEGエンコーダ42、43は、CPIを伝送する機能を有することにより、様々なアプリケーションを符号化して伝送することができる。

【0059】MPEGデコーダ44は、MPEG符号化されたエレメンタリストリーム (ES) に挿入されているCPIを分離、解釈することにより、様々なCoding Phaseを有するアプリケーションの画像を、適切に、有効画素エリアに配置するように復号処理することができる。

【0060】なお、本発明に実施の形態においては、CPIを、MPEG符号化されたエレメンタリストリームのuser dataに挿入したが、ビットストリームに他の方法で挿入するようにしてもよい。

【0061】次に、MPEGエンコーダ42において、各フィールドごとの複数の補助データを識別する動作について、図19を参照して説明する。3-2プルダウン処理が実施されている符号化フレームの原信号(30Hz)の各フィールドに、補助データが挿入されているものとする。この補助データを有する原信号(30Hz)が、元のフレームレートの符号化フレーム(24Hz)に符号化されるとき、各符号化フレームに含まれる2つまたは3つ

のフィールドに記述されている補助データに、符号化されたフィールドに対応した識別子が、field\_ID(0乃至2のカウンタの値)として付加されて(図15のエンコードコントローラ102で補助データにfield\_IDが付加されて)、補助データとともに伝送される。このfield\_IDが補助データに付加されることにより、補助データがどの符号化フレーム内の、どのフィールドに対応したものであるかが識別される。

【0062】具体的に説明すると、図19の先頭の符号化フレームのフィールドの枚数は、2枚であるので、補助データは2つ存在する。符号化フレームは2枚のフィールドから生成され、その符号化フレームに対応する2枚のフィールドの、それぞれの補助データに、field\_IDとして、“0”または“1”が付加される。

【0063】先頭から2番目の符号化フレームのフィールドの枚数は、3枚であるので、補助データは3つ存在する。符号化フレームは3枚のフィールドから生成され、その符号化フレームに対応する3枚のフィールドの、それぞれの補助データに、field\_IDとして、“0”、“1”または“2”が付加される。

【0064】つまり、1枚の符号化フレームに対して、1つの補助データが生成されるのではなく、符号化フレームに含まれていたフィールドの枚数と同じ数の補助データが生成され、それぞれにfield\_IDが付加される。その結果、同一の符号化フレームに含まれる複数の補助データは、付加されたfield\_IDにより、符号化フレーム内で識別されるので、それぞれの補助データに異なった情報が含まれていても、識別できなくなることはない。

【0065】MPEGエンコーダ42では、画像データがMPEGエンコード部103でエンコードされ、可変長符号化部104で可変長符号化される。また、エンコードコントローラ102で補助データにfield\_IDと、補助データが挿入されていたライン番号(Line\_number)が付加され、エレメンタリストリーム中のuser data中に、ancillary dataとして挿入される。これにより、複数の補助データを識別して伝送することができる。

【0066】MPEGデコーダ44では、MPEG符号化されたエレメンタリストリーム中のuser dataが可変長復号部112で分離され、デコードコントローラ113に供給される。デコードコントローラ113は、user dataに挿入されているancillary dataのfield\_IDとLine\_numberに基づいて、複数の補助データを識別、分離し、合成部115に出力する。合成部115は、MPEGデコード部114で復号された画像データと、それに対応する補助データ(テキストデータ)を合成し、出力する。

【0067】MPEGエンコーダ42は、システム全体の遅延を少なくするために、図20に示すような符号化順序と表示順序を管理するPOIを生成する。

【0068】例えば、MPEGエンコーダ42のMPEGエンコード部103に入力された画像データが、3-2プルダ



ウン処理により、24Hzのフレームレートに変換された画像信号である場合、図20(A)に示すようなフラグ(Repeat\_first\_field, Top\_field\_first)により、各フレームが管理される。

【0069】Repeat\_first\_fieldのフラグの"1"は、リピートフィールドを作成する必要があることを意味し、Repeat\_first\_fieldのフラグの"0"は、リピートフィールドを作成する必要があることを意味する。Top\_field\_firstのフラグは、フレームを構成するフィールドのうち、最初のフィールドがトップフィールドであるのか、またはボトムフィールドであるのかを表している。Top\_field\_firstフラグの"1"は、トップフィールドがボトムフィールドより時間的に早いフレーム構造であることを表しており、Top\_field\_firstフラグの"0"は、ボトムフィールドがトップフィールドより時間的に早いフレーム構造であることを表している。

【0070】図20(A)について、具体的に説明すると、最初にMPEGエンコード部103に入力されるFrame No1の符号化フレームの符号化画像データ種別は、Iピクチャであり、このIピクチャの2フィールド(トップフィールドとボトムフィールド)は、トップフィールドをコピーしてリピートフィールドを作成することで、3フィールドに変換する必要があるため、対応するRepeat\_first\_fieldのフラグは"1"となり、Top\_field\_firstのフラグは"1"となる。

【0071】Frame No2の符号化フレームの符号化画像データ種別は、Bピクチャであり、このBピクチャには、リピートフィールドが生成する必要がないので、Repeat\_first\_fieldのフラグは"0"とされ、ボトムフィールドがトップフィールドより時間的に早いフレームであるため、Top\_field\_firstのフラグは"0"とされる。このときのTop\_field\_firstのフラグの値は、3-2プルダウン処理には関係しない。

【0072】Frame No3の符号化フレームの符号化画像データ種別は、Bピクチャであり、Frame No3のBピクチャでは、そのボトムフィールドをコピーしてリピートフィールドが作成され、符号化フレームが3フィールドに変換されている。従って、Repeat\_first\_fieldのフラグは"1"とされ、Top\_field\_firstのフラグは"0"とされる。

【0073】Frame No4の符号化フレームの符号化画像データ種別は、Pピクチャであり、このPピクチャに対しては、リピートフィールドが作成されておらず、Repeat\_first\_fieldのフラグは"0"とされ、Top\_field\_firstのフラグは1とされる。

【0074】MPEGエンコード部103は、図20(A)に示すような3-2プルダウン処理が施された画像データが入力されてきたとき、内蔵するカウンタPTS\_counterでフィールドの数をカウントし、その値PTS\_counterを表示順序としてエンコードコントローラ102に出力す

る。カウンタPTS\_counterは、0から127まで増加した後、再び0に戻るカウント動作を行う。従って、カウンタPTS\_counterの値は、図20(B)に示すように変化する。

【0075】具体的に説明すると、最初に入力されるFrame No1のIピクチャのPTS\_counterの値は値"0"である。先頭から2番目に入力されるFrame No2のBピクチャのPTS\_counterの値は、Frame No1のIピクチャのPTS\_counterの値"0"に、Pピクチャのフィールド数3を加算した値"3"( $=0+3$ )となる。

【0076】先頭から3番目に入力されるFrame No3のBピクチャのPTS\_counterの値は、Frame No2のBピクチャのPTS\_counterの値"3"に、Bピクチャのフィールド数2を加算した値"5"( $=3+2$ )となる。先頭から4番目に入力されるFrame No4のPピクチャのPTS\_counterの値は、Frame No3のBピクチャのPTS\_counterの値"5"に、Bピクチャのフィールド数3を加算した値"8"( $=5+3$ )となる。Frame No5のBピクチャ以降のPTS\_counterの値も同様に算出される。

【0077】さらに、MPEGエンコード部103は、内蔵するカウンタDTS\_counterでエンコードしたフレームを計数し、計数した結果をエンドコントローラ102に出力する。

【0078】図20(C)を参照して、具体的に説明すると、Frame No1のIピクチャのDTS\_counterの値125は、Frame No1のIピクチャが表示される表示順序PTS\_counter=0を基準としたとき、1フレーム分の出現周期前に符号化される必要がある(図14に対応させると、先頭のIピクチャの符号化順序の値は"1"であり、表示順序の値は"2"であり、符号化順序の値は表示順序の値より1フレーム分早い必要がある)。つまり、Iピクチャが3つのフィールドを持っているため、DTS\_counterの値は、0より3だけ前の値"125"(DTS\_counterは $2^7 (=128)$ のモジュロで表されるため、その値は0から127の間の値を循環する)となる。

【0079】Frame No1のIピクチャの次に符号化されるFrame No4のPピクチャのDTS\_counterの値は、Frame No1のIピクチャのDTS\_counterの値125にIピクチャのフィールド数3を加えた値0( $=128=125+3$ )となる。

【0080】Frame No4のPピクチャの次に符号化される、Frame No2のBピクチャのDTS\_counterの値は、BピクチャのためにPTS\_counter=DTS\_counterであり、PTS\_counterの値と同一とされ、その値は"3"となる。同様に、Frame No2のBピクチャの次に符号化される、Frame No3のBピクチャのDTS\_counterの値も、PTS\_counterの値と同一とされ、その値は"5"とされる。以下、Frame No7のPピクチャ以降のDTS\_counterの値も、同様に算出されるので、ここでは、その説明を省略

する。

【0081】MPEGエンコード部103は、図20に示すようなフラグRepeat\_first\_field、Top\_field\_first、並びにカウンタPTS\_counter、DTS\_counterをPOIとして、エンコードコントローラ102に出力する。

【0082】ここで、図11に示した、離れた場所にあるスタジオ41のMPEGエンコーダ42から、スタジオ71のMPEGデコーダ74に、SDTI-CPネットワーク50、80、TS MUX/DEMUX 61、62、およびATMのネットワークを使用して、画像信号の伝送を行うシステムの、符号化順序と表示順序について、図21を参照して説明する。

【0083】MPEGエンコーダ42内のMPEGビデオエンコーダ42Aは、MPEG符号化された画像データのエレメンタリストリーム(ES)を出力するとともに、POIを、そのエレメンタリストリーム(ES)中のuser dataに挿入する(図15のエンコードコントローラ102がPOIのデータをuser dataに記述し、可変長符号化部104に出力して、多重化させる)。MPEGエンコーダ42内のオーディオエンコーダ42Bは、音声データを符号化してエレメンタリストリーム(ES)として出力する。SDTI-CPインタフェース46は、MPEGビデオエンコーダ42Aからのエレメンタリストリーム(ES)(POIを含んだストリーム)と、オーディオエンコーダ42Bからのエレメンタリストリーム(ES)を、SDTIフォーマットのエレメンタリストリームに変換して、SDTI-CPネットワーク50に出力する。

【0084】SDTI-CPインタフェース51は、入力されたSDTIフォーマットのエレメンタリストリーム(ES)を、MPEG符号化のエレメンタリストリーム(ES)に変換し(図18を参照してSDTI-CPインタフェース46の動作として説明したように)、TSMUX/DEMUX 61に出力する。TS MUX/DEMUX 61は、エレメンタリストリーム(ES)に挿入されているPOIを参照して、PTS\_counterの値をPTS(Presentation TimeStamp)に、また、DTS\_counterの値をDTS(Decoding Time Stamp)に、それぞれ換算し、多重化処理を行い、188バイト単位のパケットから構成されるトランスポートストリーム(TS)を生成して出力する(従って、トランスポートストリーム(TS)には、PTS\_counterとDTS\_counterではなく、PTSとDTSが含まれる)。

【0085】TS MUX/DEMUX 61は、エレメンタリストリーム(ES)に挿入されているPOIを解釈することにより、前述のバッファリング処理を行うことなく、直ちに多重化処理を行うことができ、TS MUX/DEMUX 61において、新たに遅延が発生することはない。また、POIはエレメンタリストリーム(ES)中に挿入されているため、TS MUX/DEMUX 61は、POIを後段に伝達するために、POIをビットストリームに含ませる処理を行わなくてもよい。

【0086】図11のTS MUX/DEMUX 62は、ATMなどの公衆網を介して入力されたトランスポートストリーム

(TS)から画像データと音声データを分離して、MPEG符号化のエレメンタリストリーム(ES)に変換する。TS MUX/DEMUX 62はまた、PTS、DTSをPTS\_counter、DTS\_counterに換算して、エレメンタリストリームのuser dataに挿入し、SDTI-CPインタフェース81に出力する。SDTI-CPインタフェース81は、MPEG符号化されたエレメンタリストリーム(ES)をSDTIフォーマットのエレメンタリストリーム(ES)に変換し、スタジオ71のSDTI-CPネットワーク80に出力する。MPEGデコーダ74(MPEGデコーダ44と同一の構成)は、SDTI-CPインタフェース78を介して、伝送されてきた画像データと音声データのエレメンタリストリーム(ES)を受信し、復号する。

【0087】MPEGデコーダ74は、TS MUX/DEMUX 61と同様に、エレメンタリストリーム(ES)に挿入されているPOIを解釈して、前述のバッファリング処理を行うことなく、直ちに復号する(図16のデコードコントローラ113が出力するPOIに基づき、MPEGデコード部114が画像データを復号する)ことができ、MPEGデコーダ74において、新たに遅延が発生することはない。つまり、MPEGビデオエンコーダ42Aが、MPEG符号化されたエレメンタリストリーム(ES)とともに、POIをエレメンタリストリームに挿入して出力することにより、後段のTS MUX/DEMUX 61と、MPEGデコーダ74は、POIを解釈して多重化処理、または復号処理を直ちに行うことができ、システム全体としての遅延を、MPEGエンコーダ42で生ずるBピクチャの枚数(=2)+1枚=3の遅延のみに抑えることができる。すなわち、このような符号化、多重化、および復号を含むシステムにおいて、理論的に最も小さい遅延とすることができる。

【0088】また、図21のシステム構成において、図18に示したSDTI-CPインタフェース46を用いることにより、SDTI-CPネットワーク50を使用して画像信号を伝送するとき、スタジオ41の内部では、ビットストリームを、編集が容易で、短距離伝送に適したエレメンタリストリーム(ES)の形態で伝送することが可能となり、離れたスタジオ間でATMなどの公衆網を使用して画像信号を伝送するとき、ビットストリームを、長距離伝送に適したトランスポートストリーム(TS)の形態で伝送することが可能となる。

【0089】以上においては、POIをエレメンタリストリームに挿入するようにしたが、TSMUX/DEMUX 61とMPEGエンコーダ42の距離が近いような場合には、図22に示すように、POIをMPEGエンコーダ42から、TS MUX/DEMUX 61に直接供給するようにしてもよい。

【0090】しかしながら、このようにすると、エレメンタリストリームを伝送するSDTI-CPネットワーク50以外の配線処理が必要となる。

【0091】以上に述べたように、エンコーダの有する情報を、エレメンタリストリーム(ES)中のuser dataに記述して、多重化装置またはデコーダに出力することにより、エンコーダのみが有していた情報(Coding Phase(V-PhaseとH-Phase)、Field\_ID、符号化順序DTS\_counter、および表示順序PTS\_counter)を、エンコーダより後段の多重化装置、デコーダに供給することができる。

【0092】次に、図23乃至図36を参照してビットストリームのシンタックスについて説明する。

【0093】図23は、MPEGのビデオストリームのシンタックスを表わした図である。MPEGエンコーダ42は、この図23に示されたシンタックスに従った符号化エレメンタリストリームを生成する。以下に説明するシンタックスにおいて、関数や条件文は細活字で表わされ、データエレメントは、太活字で表されている。データ項目は、その名称、ビット長およびそのタイプ・伝送順序を示すニーモニック(Mnemonic)で記述されている。

【0094】まず、この図23に示されているシンタックスにおいて使用されている関数について説明する。実際には、この図23に示されているシンタックスは、MP  
EGデコーダ44側において、伝送されてきた符号化ビットストリームから所定の意味のあるデータエレメントを抽出するために使用されるシンタックスである。MPEGエンコーダ42側において使用されるシンタックスは、図23に示されたシンタックスからif文やwhile文等の条件文を省略したシンタックスである。

【0095】video\_sequence()において最初に記述されているnext\_start\_code()関数は、ビットストリーム中に記述されているスタートコードを探すための関数である。この図23に示されたシンタックスに従って生成された符号化ストリームには、まず最初に、sequence\_header()関数とsequence\_extension()関数によって定義されたデータエレメントが記述されている。このsequence\_header()関数は、MPEGビットストリームのシーケンスレイヤのヘッダデータを定義するための関数であって、sequence\_extension()関数は、MPEGビットストリームのシーケンスレイヤの拡張データを定義するための関数である。

【0096】sequence\_extension()関数の次に配置されているdo{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントが符号化データストリーム中に記述されていることを示す構文である。このwhile文に使用されているnextbits()関数は、ビットストリーム中に記述されているビット又はビット列と、参照されるデータエレメントとを比較するための関数である。この図23に示されたシンタックスの例では、nextbits()関数は、ビットストリーム中のビット列とビデオシーケンスの終わりを示すsequence\_end\_codeとを比較し、ビッ

ットストリーム中のビット列とsequence\_end\_codeとが一致しないときに、このwhile文の条件が真となる。従って、sequence\_extension()関数の次に配置されているdo{}while構文は、ビットストリーム中に、ビデオシーケンスの終わりを示すsequence\_end\_codeが現れない間、do文中の関数によって定義されたデータエレメントが符号化ビットストリーム中に記述されていることを示している。

【0097】符号化ビットストリームにおいて、sequence\_extension()関数によって定義された各データエレメントの次には、extension\_and\_user\_data(0)関数によって定義されたデータエレメントが記述されている。このextension\_and\_user\_data(0)関数は、MPEGビットストリームのシーケンスレイヤにおける拡張データとユーザデータを定義するための関数である。

【0098】このextension\_and\_user\_data(0)関数の次に配置されているdo{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントが、ビットストリーム中に記述されていることを示す関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、picture\_start\_code又はgroup\_start\_codeとの一致を判断するための関数である。従って、ビットストリーム中に現れるビット又はビット列と、picture\_start\_code又はgroup\_start\_codeとが一致する場合には、while文によって定義された条件が真となる。よって、このdo{}while構文は、符号化ビットストリーム中において、picture\_start\_code又はgroup\_start\_codeが現れた場合には、そのスタートコードの次に、do文中の関数によって定義されたデータエレメントのコードが記述されていることを示している。

【0099】このdo文の最初に記述されているif文は、符号化ビットストリーム中にgroup\_start\_codeが現れた場合、という条件を示している。このif文による条件は真である場合には、符号化ビットストリーム中には、このgroup\_start\_codeの次にgroup\_of\_picture\_header()関数およびextension\_and\_user\_data(1)関数によって定義されているデータエレメントが順に記述されている。

【0100】このgroup\_of\_picture\_header()関数は、MPEG符号化ビットストリームのGOPレイヤのヘッダデータを定義するための関数であって、extension\_and\_user\_data(1)関数は、MPEG符号化ビットストリームのGOPレイヤの拡張データおよびユーザデータを定義するための関数である。

【0101】さらに、この符号化ビットストリームにおいて、group\_of\_picture\_header()関数およびextension\_and\_user\_data(1)関数によって定義されているデータエレメントの次には、picture\_header()関数とpicture\_coding\_extension()関数によって定義されたデータエレ

10

20

30

40

50

メントが記述されている。もちろん、先に説明したif文の条件が真とならない場合には、group\_of\_picture\_header()関数およびextension\_and\_user\_data(1)関数によって定義されているデータエレメントは記述されていないので、extension\_and\_user\_data(0)関数によって定義されているデータエレメントの次に、picture\_header()関数、picture\_coding\_extension()関数およびextension\_and\_user\_data(2)関数によって定義されたデータエレメントが記述されている。

【0102】このpicture\_header()関数は、MPEG符号化ビットストリームのピクチャレイヤのヘッダデータを定義するための関数であって、picture\_coding\_extension()関数は、MPEG符号化ビットストリームのピクチャレイヤの第1の拡張データを定義するための関数である。extension\_and\_user\_data(2)関数は、MPEG符号化ビットストリームのピクチャレイヤの拡張データおよびユーザデータを定義するための関数である。このextension\_and\_user\_data(2)関数によって定義されるユーザデータは、ピクチャレイヤに記述されているデータであって、各ピクチャ毎に記述することのできるデータである。

【0103】符号化ビットストリームにおいて、ピクチャレイヤのユーザデータの次には、picture\_data()関数によって定義されるデータエレメントが記述されている。このpicture\_data()関数は、スライスレイヤおよびマクロブロックレイヤに関するデータエレメントを記述するための関数である。

【0104】このpicture\_data()関数の次に記述されているwhile文は、このwhile文によって定義されている条件が真である間、次のif文の条件判断を行うための関数である。このwhile文において使用されているnextbits()関数は、符号化ビットストリーム中に、picture\_start\_code又はgroup\_start\_codeが記述されているか否かを判断するための関数であって、ビットストリーム中にpicture\_start\_code又はgroup\_start\_codeが記述されている場合には、このwhile文によって定義された条件が真となる。

【0105】次のif文は、符号化ビットストリーム中にsequence\_end\_codeが記述されているか否かを判断するための条件文であって、sequence\_end\_codeが記述されていないのであれば、sequence\_header()関数とsequence\_extension()関数とによって定義されたデータエレメントが記述されていることを示している。sequence\_end\_codeは符号化ビデオストリームのシーケンスの終わりを示すコードであるので、符号化ストリームが終了しない限り、符号化ストリーム中にはsequence\_header()関数とsequence\_extension()関数とによって定義されたデータエレメントが記述されている。

【0106】このsequence\_header()関数とsequence\_extension()関数によって記述されたデータエレメントは、ビデオストリームのシーケンスの先頭に記述された

sequence\_header()関数とsequence\_extension()関数によって記述されたデータエレメントと全く同じである。このように同じデータをストリーム中に記述する理由は、ビットストリーム受信装置側でデータストリームの途中（例えばピクチャレイヤに対応するビットストリーム部分）から受信が開始された場合に、シーケンスレイヤのデータを受信できなくなり、ストリームをデコード出来なくなることを防止するためである。

【0107】この最後のsequence\_header()関数とsequence\_extension()関数とによって定義されたデータエレメントの次、つまり、データストリームの最後には、シーケンスの終わりを示す2ビットのsequence\_end\_codeが記述されている。

【0108】以下に、sequence\_header()関数、sequence\_extension()関数、extension\_and\_user\_data(0)関数、group\_of\_picture\_header()関数、picture\_header()関数、picture\_coding\_extension()関数、およびpicture\_data()関数について詳細に説明する。

【0109】図24は、sequence\_header()関数のシンタックスを説明するための図である。このsequence\_header()関数によって定義されたデータエレメントは、sequence\_header\_code、horizontal\_size\_value、vertical\_size\_value、aspect\_ratio\_information、frame\_rate\_code、bit\_rate\_value、marker\_bit、vbv\_buffer\_size\_value、constrained\_parameter\_flag、load\_intra\_quantizer\_matrix、intra\_quantizer\_matrix[64]、load\_non\_intra\_quantizer\_matrix、およびnon\_intra\_quantizer\_matrix等である。

【0110】sequence\_header\_codeは、シーケンスレイヤのスタート同期コードを表すデータである。horizontal\_size\_valueは、画像の水平方向の画素数の下位12ビットから成るデータである。vertical\_size\_valueは、画像の縦のライン数の下位12ビットからなるデータである。aspect\_ratio\_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame\_rate\_codeは、画像の表示周期を表すデータである。bit\_rate\_valueは、発生ビット量に対する制限のためのビット・レートの下位18ビット（4000bsp単位で切り上げる）データである。marker\_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。vbv\_buffer\_size\_valueは、発生符号量制御用の仮想バッファ（ビデオバッファリファイヤー）の大きさを決める値の下位10ビットデータである。constrained\_parameter\_flagは、各パラメータが制限以内であることを示すデータである。load\_intra\_quantizer\_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra\_quantizer\_matrix[64]は、イントラMB用量子化マトリックスの値を示すデータである。load\_non\_intra\_quantizer\_matrixは、非イントラMB用量子化マトリックス・デ

ータの存在を示すデータである。non\_intra\_quantizer\_matrixは、非イントラMB用量子化マトリックスの値を表すデータである。

【0111】図25はsequence\_extension()関数のシンタックスを説明するための図である。このsequence\_extension()関数によって定義されたデータエレメントとは、extension\_start\_code、extension\_start\_code\_identifier、profile\_and\_level\_indication、progressive\_sequence、chroma\_format、horizontal\_size\_extension、vertical\_size\_extension、bit\_rate\_extension、vbv\_buffer\_size\_extension、low\_delay、frame\_rate\_extension\_n、および frame\_rate\_extension\_d等のデータエレメントである。

【0112】extension\_start\_codeは、エクステンションデータのスタート同期コードを表すデータである。extension\_start\_code\_identifierは、どの拡張データが送られるかを示すデータである。profile\_and\_level\_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive\_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma\_formatは、ビデオデータの色差フォーマットを指定するためのデータである。horizontal\_size\_extensionは、シーケンスヘッダのhorizontal\_size\_valueに加える上位2ビットのデータである。vertical\_size\_extensionは、シーケンスヘッダのvertical\_size\_valueに加える上位2ビットのデータである。bit\_rate\_extensionは、シーケンスヘッダのbit\_rate\_valueに加える上位12ビットのデータである。vbv\_buffer\_size\_extensionは、シーケンスヘッダのvbv\_buffer\_size\_valueに加える上位8ビットのデータである。low\_delayは、Bピクチャを含まないことを示すデータである。frame\_rate\_extension\_nは、シーケンスヘッダのframe\_rate\_codeと組み合わせてフレームレートを取得するためのデータである。frame\_rate\_extension\_dは、シーケンスヘッダのframe\_rate\_codeと組み合わせてフレームレートを取得するためのデータである。

【0113】図26は、extension\_and\_user\_data(i)関数のシンタックスを説明するための図である。このextension\_and\_user\_data(i)関数は、「i」が1以外ときは、extension\_data()関数によって定義されるデータエレメントは記述せずに、user\_data()関数によって定義されるデータエレメントのみを記述する。よって、extension\_and\_user\_data(0)関数は、user\_data()関数によって定義されるデータエレメントのみを記述する。

【0114】まず、図26に示されているシンタックスにおいて使用されている関数について説明する。nextbits()関数は、ビットストリーム中に現れるビットまたはビット列と、次に復号されるデータエレメントとを比較するための関数である。

【0115】user\_data()関数は、図27に示すよう

に、user\_data\_start\_code、V-phase()関数、H-phase()関数、Time\_code()関数、Picture-order()関数、Ancillary\_data()関数、history\_data()関数、およびuser\_dataのデータエレメントを記述するための関数である。

【0116】user\_data\_start\_codeは、MPEG方式のビットストリームのピクチャレイヤのユーザデータエリアの開始を示すためのスタートコードである。このuser\_data\_start\_codeの次に記述されているif文は、user\_data(i)関数のiが"0"のとき、次に記述されているwhile構文を実行する。このwhile構文は、ビットストリーム中に、23個の"0"とそれに続く"1"から構成される24ビットのデータが現れない限り真となる。

【0117】この23個の"0"とそれに続く"1"から構成される24ビットのデータは、すべてのスタートコードの先頭に付与されるデータであって、すべてのスタートコードは、この24ビットの後ろに設けられることによって、nextbits()関数は、ビットストリーム中において、各スタートコードの位置を見つけることができる。

【0118】while構文が真のとき、その次に記述されているif文のnextbits()関数は、V-Phaseを示すビット列(Data\_ID)を検出すると、そのビット列(Data\_ID)の次ビットからV-Phase()関数で示されるV-Phaseのデータエレメントが記述されていることを知る。次のElse if文のnextbits()関数は、H-Phaseを示すビット列(Data\_ID)を検出すると、そのビット列(Data\_ID)の次ビットからH-Phase()関数で示されるH-Phaseのデータエレメントが記述されていることを知る。

【0119】ここで、図28に示すように、V-PhaseのData\_IDは、"01"を表すビット列であり、H-PhaseのData\_IDは、"02"を表すビット列である。

【0120】ビットストリームに記述されるV-Phase()関数のシンタックスについて、図29を参照して説明する。まず、Data\_IDは、前述したように、そのData\_IDの次のビット列のデータエレメントがV-Phaseであることを表す8ビットのデータであり、図28で示した値"01"である。V-Phaseは、画像信号のフレームにおいて、符号化される最初のラインを示す16ビットのデータである。

【0121】ビットストリームに記述されるH-Phase()関数のシンタックスについて、図30を参照して説明する。まず、Data\_IDは、前述したように、そのData\_IDの次のビット列のデータエレメントがH-Phaseであることを表す8ビットのデータであり、図28で示した値"02"である。H-Phaseは、画像信号フレームにおいて、符号化される最初のサンプルを示す8ビットのデータである。

【0122】図27に戻って、次のElse if文は、user\_data(i)関数のiが2のとき、次に記述されているwhile構文を実行する。while構文の内容は前述した場合と同

様であるので、ここではその説明を省略する。

【0123】while構文が真のとき、次のif文において、nextbits()関数は、Time code1を示すビット列を検出するか、または、Time code2を示すビット列を検出すると、そのビット列の次ビットからTime\_code()関数で示されるTime codeのデータエレメントが記述されていることを知る。

【0124】Time code1のData\_IDは、図28に示すように、“03”を表すビット列であり、Time code1のデータは、画像の垂直ブランキング期間に挿入されたタイムコードを示す、VITC (Vertical Interval Time Code) である。Time code2のData\_IDは、図28に示すように、“04”を表すビット列であり、Time code2のデータは、記録媒体のタイムコードトラックに記録されたタイムコードを示す、LTC (Longitudinal Time Code) である。

【0125】次に、Else if文において、nextbits()関数は、Picture Orderを示すビット列を検出すると、そのビット列の次ビットからPicture\_Order()関数で示されるPicture Orderのデータエレメントが記述されていることを知る。ここで、Picture\_Order()関数のData\_IDは、図28に示すように、“05”を表すビット列である。

【0126】実際に、エンコーダでエレメンタリストリーム (ES) に挿入するPicture\_Order()関数のシンタックスを、図31を参照して説明する。まず、Data\_IDは前述したように、そのData\_ID以降のデータがPOIのデータであることを示す8ビットのデータであり、その値は“05”である。DTS\_presenceは、符号化順序DTS\_counterの有無を表す1ビットのデータである。例えば、BピクチャのようにDTS\_counter=PTS\_counterとなる場合、表示順序PTS\_counterのみが存在し、DTS\_presenceのビットは“0”となる。逆に、PピクチャおよびIピクチャの場合、符号化順序DTS\_counterと表示順序PTS\_counterは同一ではないので、表示順序PTS\_counterと符号化順序DTS\_counterの双方が存在し、DTS\_presenceのビットは1となる。

【0127】PTS\_counterは、エンコーダに符号化フレーム中の1フィールドが入力されるごとにカウントアップを行う、表示順序を表す7ビットのデータである。この7ビットのデータは、0から127までの値をとるモジュロである。if文以降は、DTS\_presenceのビットが1のとき、すなわち、PピクチャおよびIピクチャのとき、DTS\_counterのカウントアップが実行される。

【0128】Marker\_bitsは、user dataの記述されたビット列が、偶然に前述したスタートコードと一致し、画像破綻を引き起こす可能正が高い、スタートコードエミュレーションを防止するために、16ビットごとに挿入されるビットである。

【0129】DTS\_counterは、エンコーダで、1フィー

ルド分の符号化画像データが符号化されるごとにカウントアップを行う、符号化順序を表す7ビットのデータである。この7ビットのデータは、0から127までの値をとるモジュロである。

【0130】前述したように、表示順序PTS\_counterは、フィールド単位で付番されるために、例えば、符号化画像データを24Hzから30Hzのフレームレートに変換して符号化する場合、3-2プルダウン処理を行った後に、付番する必要がある。

【0131】図27に戻って、その次に記述されているwhile構文も、内容は前述した場合と同様であるので、ここではその説明を省略する。while構文が真のとき、次のif文において、nextbits()関数は、Ancillary dataを示すビット列を検出すると、そのビット列の次ビットからAncillary\_data()関数で示されるAncillary dataのデータエレメントが記述されていることを知る。Ancillary\_data()関数のData\_IDは、図28に示すように、“07”を表すビット列である。

【0132】この補助データに識別子を付加するancillary dataのシンタックスを図32を参照して説明する。Ancillary\_data()関数はピクチャ層のuser dataとして伝送され、データとしてはField識別子 (Field\_ID)、ラインの番号 (Line\_number) および補助データ (ancillary data) が挿入される。

【0133】Data\_IDは、user data領域において、ancillary dataであることを示す8ビットのデータであり、その値は図28に示したように“07”である。

【0134】Field\_IDは2ビットのデータであり、progressive\_sequence flag (図25) の値が“0”のとき、符号化フレーム内のフィールドごとにField\_IDが付加される。repeat\_first\_fieldに“0”が設定されているとき、この符号化フレームにはフィールドが2枚存在し、Field\_IDは、図19に示したように、最初のフィールドに“0”、およびその次のフィールドに“1”が設定され、repeat\_first\_fieldに“1”が設定されているとき、この符号化フレームにはフィールドが3枚存在し、Field\_IDとしては、最初のフィールドに“0”が設定され、それ以降のフィールドに“1”、“2”が設定される。

【0135】Field\_IDは、progressive\_sequence flagの値が“1”のとき、符号化フレームごとに付加される。Field\_IDには、repeat\_first\_fieldとTop\_field\_firstにともに“0”が設定されているとき、その符号化フレームは1枚のprogressive frameが存在するので、値“0”が設定され、repeat\_first\_fieldに値“1”およびTop\_field\_firstに値“0”が設定されているとき、その符号化フレームは2枚のprogressive frameが存在するので、値“0”、“1”が設定され、repeat\_first\_fieldとTop\_field\_firstにともに“1”が設定されているとき、その符号化フレームは3枚のprogressiv

10

20

30

40

50

e frameが存在するので、値“0”乃至“2”が設定される。

【0136】Line\_numberは、14ビットのデータであり、各フレームにおける補助データが記述されている、ITU-R BT.656-3, SMPTE274M, SMPTE293M, SMPTE296Mで規定されたライン番号を示す。

【0137】Ancillary\_data\_lengthは、16ビットのデータであり、ancillary\_data\_payloadのデータ長を示す。Ancillary\_data\_payloadは、22ビットのデータからなる補助データの内容を表しており、Ancillary\_data\_payloadのAncillary\_data\_lengthの値がjの値(初期値0)より大きいとき、値j(Ancillary\_data\_lengthのデータ長)を1だけインクリメントして、そのjの値のビット列目から記述される。

【0138】次のWhile構文は、bytealigned()関数のためのシンタックスを表しており、次のデータがbytealigned()関数でないとき(While構文が真のとき)、Zero\_bit(1ビットのデータ“0”)を記述する。

【0139】図27に戻って、次のElse if文において、nextbits()関数は、History dataを示すビット列を検出すると、そのビット列の次ビットからHistory\_data()関数で示されるHistory dataのデータエレメントが記述されていることを知る。History\_data()関数のData\_IDは、図28に示すように、“08”を表すビット列であり、Data\_IDが“08”で示されるデータは、符号化パラメータの履歴情報を含むHistory dataを表している。

【0140】最後のif文において、nextbits()関数は、user dataを示すビット列を検出すると、そのビット列の次ビットからuser\_data()関数で示されるuser\_dataのデータエレメントが記述されていることを知る。

【0141】図27のnextbits()関数が、それぞれのデータエレメントが記述されていることを知るビット列は、図28に示すData\_IDとして記述されている。ただし、Data\_IDとして“00”を使用することは禁止されている。Data\_IDが“80”で示されるデータは、制御フラグを表しており、Data\_IDが“FF”で示されるデータは、user dataを表している。

【0142】図33は、group\_of\_picture\_header()関数のシンタックスを説明するための図である。このgroup\_of\_picture\_header()関数によって定義されたデータエレメントは、group\_start\_code、time\_code、closed\_gop、およびbroken\_linkから構成される。

【0143】group\_start\_codeは、GOPレイヤの開始同期コードを示すデータである。time\_codeは、GOPの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed\_gopは、GOP内の画像が他のGOPから独立再生可能なことを示すフラグデータである。broken\_linkは、編集などのためにGOP内の先頭のBピクチャが正確に再生できないことを示すフラグデータである。

【0144】extension\_and\_user\_data(1)関数は、extension\_and\_user\_data(0)関数と同じように、user\_data()関数によって定義されるデータエレメントのみを記述するための関数である。

【0145】次に、図34乃至図36を参照して、符号化ストリームのピクチャレイヤに関するデータエレメントを記述するためのpicture\_headr()関数、picture\_coding\_extension()関数、およびpicture\_data()について説明する。

【0146】図34はpicture\_headr()関数のシンタックスを説明するための図である。このpicture\_headr()関数によって定義されたデータエレメントは、picture\_start\_code、temporal\_reference、picture\_coding\_type、vbv\_delay、full\_pel\_forward\_vector、forward\_f\_code、full\_pel\_backward\_vector、backward\_f\_code、extra\_bit\_picture、およびextra\_information\_pictureである。

【0147】具体的には、picture\_start\_codeは、ピクチャレイヤの開始同期コードを表すデータである。temporal\_referenceは、ピクチャの表示順を示す番号で、GOPの先頭でリセットされるデータである。picture\_coding\_typeは、ピクチャタイプを示すデータである。

【0148】vbv\_delayは、VBVバッファの初期状態を示すデータであって、各ピクチャ毎に設定されている。送信側システムから受信側システムに伝送された符号化エレメントリストのピクチャは、受信側システムに設けられたVBVバッファにバッファリングされ、DTS(Decoding Time Stamp)によって指定された時刻に、このVBVバッファから引き出され(読み出され)、デコーダに供給される。vbv\_delayによって定義される時間は、復号化対象のピクチャがVBVバッファにバッファリングされ始めてから、符号化対象のピクチャがVBVバッファから引き出されるまでの時間、つまりDTSによって指定された時刻までの時間を意味する。このピクチャヘッドに格納されたvbv\_delayを使用することによって、VBVバッファのデータ占有量が不連続にならないシームレスなスプライシングが実現できる。

【0149】full\_pel\_forward\_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward\_f\_codeは、順方向動きベクトル探索範囲を表すデータである。full\_pel\_backward\_vectorは、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward\_f\_codeは、逆方向動きベクトル探索範囲を表すデータである。extra\_bit\_pictureは、後続する追加情報の存在を示すフラグである。このextra\_bit\_pictureが「1」の場合には、次にextra\_information\_pictureが存在し、extra\_bit\_pictureが「0」の場合には、これに続くデータが無いことを示している。extra\_information\_pictureは、規格において予約された情報である。



【0150】図35は、picture\_coding\_extension()関数のシンタックスを説明するための図である。このpicture\_coding\_extension()関数によって定義されたデータエレメントは、extension\_start\_code、extension\_start\_code\_identifier、f\_code[0][0]、f\_code[0][1]、f\_code[1][0]、f\_code[1][1]、intra\_dc\_precision、picture\_structure、top\_field\_first、frame\_predictive\_frame\_dct、concealment\_motion\_vectors、q\_scale\_type、intra\_vlc\_format、alternate\_scan、repeat\_first\_field、chroma\_420\_type、progressive\_frame、composite\_display\_flag、v\_axis、field\_sequence、sub\_carrier、burst\_amplitude、およびsub\_carrier\_phaseから構成される。

【0151】extension\_start\_codeは、ピクチャレイヤのエクステンションデータのスタートを示す開始コードである。extension\_start\_code\_identifierは、どの拡張データが送られるかを示すコードである。f\_code[0][0]は、フォワード方向の水平動きベクトル探索範囲を表すデータである。f\_code[0][1]は、フォワード方向の垂直動きベクトル探索範囲を表すデータである。f\_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f\_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。intra\_dc\_precisionは、DC係数の精度を表すデータである。picture\_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。これは、フィールドストラクチャの場合は、上位フィールドか下位フィールドかもあわせて示す。

【0152】top\_field\_firstは、フレームストラクチャの場合、最初のフィールドがトップフィールドであるのか、ボトムフィールドであるのかを示すフラグである。frame\_predictive\_frame\_dctは、フレーム・ストラクチャの場合、フレーム・モードDCTの予測がフレーム・モードだけであることを示すデータである。concealment\_motion\_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。q\_scale\_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra\_vlc\_formatは、イントラマクロブロックに、別の2次元VLC（可変長符号）を使うかどうかを示すデータである。alternate\_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。

【0153】repeat\_first\_fieldは、復号化時にリピートフィールドを生成するか否かを示すフラグであって、復号化時の処理において、repeat\_first\_fieldが「1」の場合にはリピートフィールドを生成し、repeat\_first\_fieldが「0」の場合にはリピートフィールドを生成しないという処理が行われる。

【0154】chroma\_420\_typeは、信号フォーマットが

4:2:0の場合、次のprogressive\_frameと同じ値、そうでない場合は0を表すデータである。progressive\_frameは、そのピクチャが、順次走査できているかどうかを示すデータである。composite\_display\_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。v\_axisは、ソース信号が、PALの場合に使われるデータである。field\_sequenceは、ソース信号が、PALの場合に使われるデータである。sub\_carrierは、ソース信号が、PALの場合に使われるデータである。burst\_amplitudeは、ソース信号が、PALの場合に使われるデータである。sub\_carrier\_phaseは、ソース信号が、PALの場合に使われるデータである。

【0155】図36は、picture\_data()関数のシンタックスを説明するための図である。このpicture\_data()関数によって定義されるデータエレメントは、slice()関数によって定義されるデータエレメントである。但し、ビットストリーム中に、slice()関数のスタートコードを示すslice\_start\_codeが存在しない場合には、このslice()関数によって定義されるデータエレメントはビットストリーム中に記述されていない。

【0156】slice()関数は、スライスレイヤに関するデータエレメントを記述するための関数であって、具体的には、slice\_start\_code、slice\_quantiser\_scale\_code、intra\_slice\_flag、intra\_slice、reserved\_bits、extra\_bit\_slice、extra\_information\_slice、およびextra\_bit\_slice等のデータエレメントと、macroblock()関数によって定義されるデータエレメントを記述するための関数である。

【0157】slice\_start\_codeは、slice()関数によって定義されるデータエレメントのスタートを示すスタートコードである。slice\_quantiser\_scale\_codeは、このスライスレイヤに存在するマクロブロックに対して設定された量子化ステップサイズを示すデータである。しかし、各マクロブロック毎に、quantiser\_scale\_codeが設定されている場合には、各マクロブロックに対して設定されたmacroblock\_quantiser\_scale\_codeのデータが優先して使用される。intra\_slice\_flagは、ビットストリーム中にintra\_sliceおよびreserved\_bitsが存在するか否かを示すフラグである。intra\_sliceは、スライスレイヤ中にノンイントラマクロブロックが存在するか否かを示すデータである。スライスレイヤにおけるマクロブロックのいずれかがノンイントラマクロブロックである場合には、intra\_sliceは「0」となり、スライスレイヤにおけるマクロブロックの全てがノンイントラマクロブロックである場合には、intra\_sliceは「1」となる。reserved\_bitsは、7ビットのデータであって「0」の値を取る。extra\_bit\_sliceは、符号化ストリームとして追加の情報が存在することを示すフラグであって、次にextra\_information\_sliceが存在する場合には「1」に設定される。追加の情報が存在しない場合に

は「0」に設定される。

【0158】macroblock()関数は、マクロブロックレイヤに関するデータエレメントを記述するための関数であって、具体的には、macroblock\_escape、macroblock\_address\_increment、およびmacroblock\_quantiser\_scale\_code等のデータエレメントと、macroblock\_modes()関数、および macroblock\_vectors(s)関数によって定義されたデータエレメントを記述するための関数である。

【0159】macroblock\_escapeは、参照マクロブロックと前のマクロブロックとの水平方向の差が34以上であるか否かを示す固定ビット列である。参照マクロブロックと前のマクロブロックとの水平方向の差が34以上の場合には、macroblock\_address\_incrementの値に33をプラスする。macroblock\_address\_incrementは、参照マクロブロックと前のマクロブロックとの水平方向の差を示すデータである。もし、このmacroblock\_address\_incrementの前にmacroblock\_escapeが1つ存在するのであれば、このmacroblock\_address\_incrementの値に33をプラスした値が、実際の参照マクロブロックと前のマクロブロックとの水平方向の差分を示すデータとなる。macroblock\_quantiser\_scale\_codeは、各マクロブロック毎に設定された量子化ステップサイズである。各スライスレイヤには、スライスレイヤの量子化ステップサイズを示すslice\_quantiser\_scale\_codeが設定されているが、参照マクロブロックに対してmacroblock\_quantiser\_scale\_codeが設定されている場合には、この量子化ステップサイズを選択する。

【0160】図37は、MPEG符号化ストリームのデータ構造を示す説明図である。この図に示したように、ビデオエレメンタリストリームのデータ構造は、少なくともシーケンスレイヤ、GOPレイヤ、およびピクチャレイヤを含んでいる。

【0161】シーケンスレイヤは、next\_start\_code()関数201、sequence\_header()関数202、extension\_start\_code()203、sequence\_extension()関数204、extension\_and\_user\_data(0)関数205によって定義されるデータエレメントから構成されている。GOPレイヤは、group\_start\_code206、group\_of\_picture\_header()関数207、extension\_and\_user\_data(1)関数208によって定義されるデータエレメントから構成されている。ピクチャレイヤは、picture\_header()関数209、picture\_coding\_extension()関数210、extension\_and\_user\_data(2)関数211、picture\_data()関数212によって定義されるデータエレメントを含んでいる。ビデオシーケンスの最後には、sequence\_end\_code213が記述されている。

【0162】extension\_and\_user\_data(2)関数211は、既に図26において説明したシンタックスからも理解できるように、user\_data\_start\_code214、user\_data()関数215、next\_start\_code216によって定義

されるデータエレメントを含んでいる。

【0163】user\_data()関数215は、既に図27において説明したシンタックスからも理解できるように、time\_code()関数217とuser\_data218によって定義されるデータエレメントを含んでいる。

【0164】なお、本明細書において、システムとは、複数の装置により構成される装置全体を表すものとする。

【0165】また、本明細書中において、上記処理を実行するコンピュータプログラムをユーザに提供する提供媒体には、磁気ディスク、CD-ROMなどの情報記録媒体の他、インターネット、デジタル衛星などのネットワークによる伝送媒体も含まれる。

【0166】

【発明の効果】以上の如く、請求項1に記載のデジタル信号伝送装置、請求項4に記載のデジタル信号伝送方法、および請求項5に記載の提供媒体によれば、第1のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第2のビットストリームに挿入するようにしたので、所定の場所における画像データの処理を迅速に行うことができるようになるとともに、他の場所においても、画像データを迅速に処理することが可能となる。

【図面の簡単な説明】

【図1】Coding Phaseを説明する図である。

【図2】補助データを説明する図である。

【図3】エレメンタリストリームを伝送するシステムの構成を示す図である。

【図4】補助データを伝送するシステムの構成を示す図である。

【図5】MPEG方式における予測構造と符号化構造を説明する図である。

【図6】MPEGエンコーダで発生する遅延について説明する図である。

【図7】スタジオの内部の構成を示すブロック図である。

【図8】図7のスタジオの内部で伝送されるデータを説明する図である。

【図9】図7のシステムにおいてデータを伝送するシステムの構成を示す図である。

【図10】3-2プルダウン処理された画像の補助データを説明する図である。

【図11】2つのスタジオの間でビットストリームを伝送するときのシステム構成を表すブロック図である。

【図12】図11のシステムで伝送されるストリームの構成を表す図である。

【図13】図11のシステムにおいて、トランスポートストリームを伝送する構成を表すブロック図である。

【図14】符号化処理の遅延を説明する図である。

【図15】本発明を適用したMPEGエンコーダの構成を表

すブロック図である。

【図 16】本発明を適用したMPEGデコーダの構成を表すブロック図である。

【図 17】CPIを伝送する場合のシステムの構成を表す図である。

【図 18】図 11のSDTI-CPインタフェース46の構成を表すブロック図である。

【図 19】補助データの伝送を説明する図である。

【図 20】3-2プルダウン処理におけるPTS\_counterとDTS\_counterを説明する図である。

【図 21】POIを伝送する場合のシステムの構成を表す図である。

【図 22】POIを伝送する場合の他のシステムの構成を表す図である。

【図 23】video\_sequence関数のシンタックスを説明する図である。

【図 24】sequence\_header()関数のシンタックスを説明する図である。

【図 25】sequence\_extension()関数のシンタックスを説明する図である。

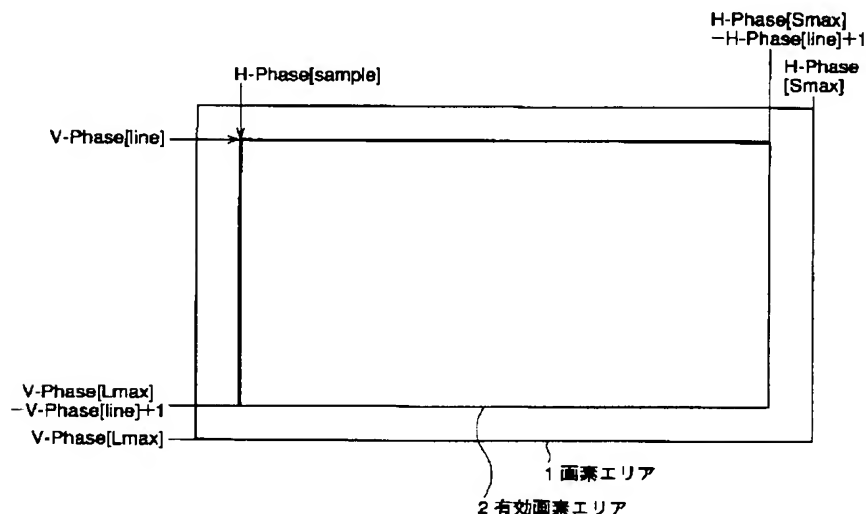
【図 26】extension\_and\_user\_data(i)関数のシンタックスを説明する図である。

【図 27】user\_data()関数のシンタックスを表す図である。

【図 28】user\_dataに記述される関数のData\_IDを説明する図である。

\*

【図 1】



\* 【図 29】V-Phase()関数のシンタックスを説明する図である。

【図 30】H-Phase()関数のシンタックスを説明する図である。

【図 31】Picture\_Order()関数のシンタックスを説明する図である。

【図 32】Ancillary\_data()関数のシンタックスを説明する図である。

【図 33】group\_of\_picture\_header()関数のシンタックスを説明する図である。

【図 34】picture\_header()関数のシンタックスを説明する図である。

【図 35】picture\_coding\_extension()関数のシンタックスを説明する図である。

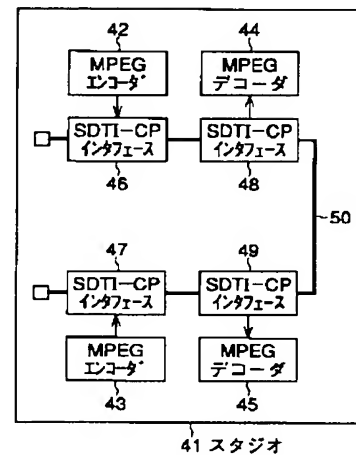
【図 36】picture\_data()関数のシンタックスを説明する図である。

【図 37】MPEG方式のレイヤを説明する図である。

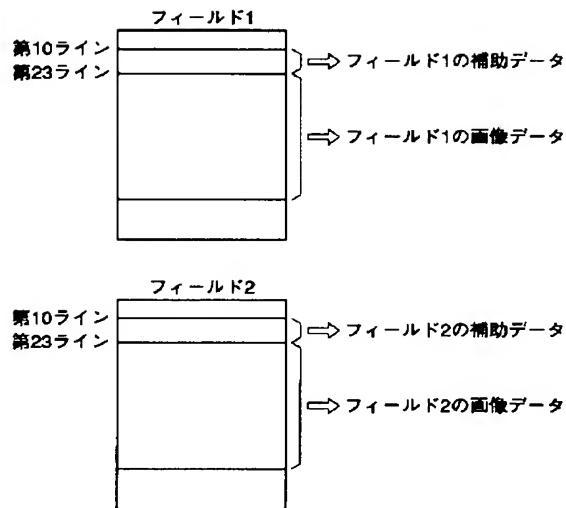
【符号の説明】

42, 43 MPEGエンコーダ, 44, 45 MPEGデコーダ, 46乃至49 SDTI-CPインタフェース, 50 SDTI-CPネットワーク, 51 SDTI-CPインタフェース, 61, 62 TS MUX/DEMUX, 72, 73 MPEGエンコーダ, 74, 75 MPEGデコーダ, 76乃至81 SDTI-CPインタフェース, 80 SDTI-CPネットワーク

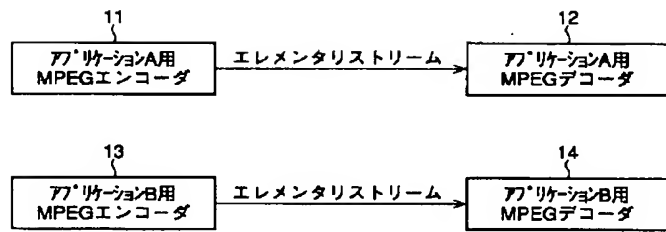
【図 7】



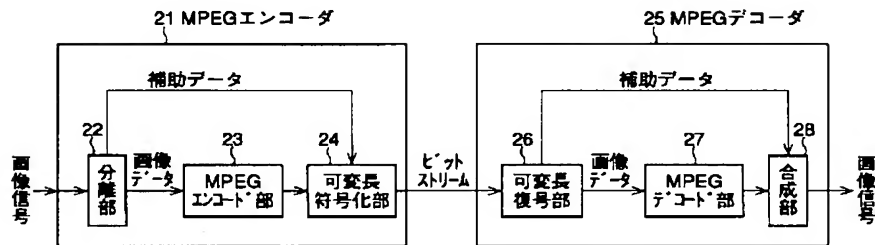
【図2】



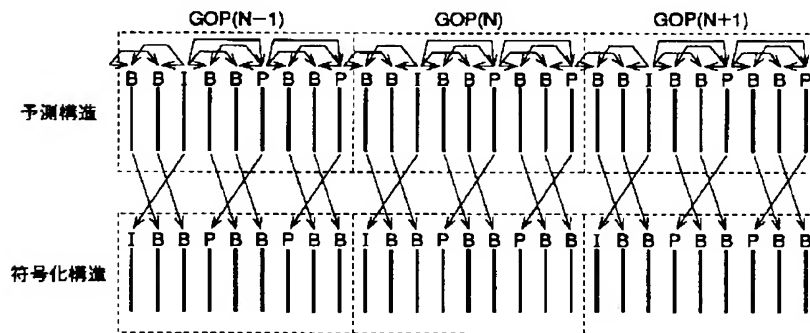
【図3】



【図4】



【図5】

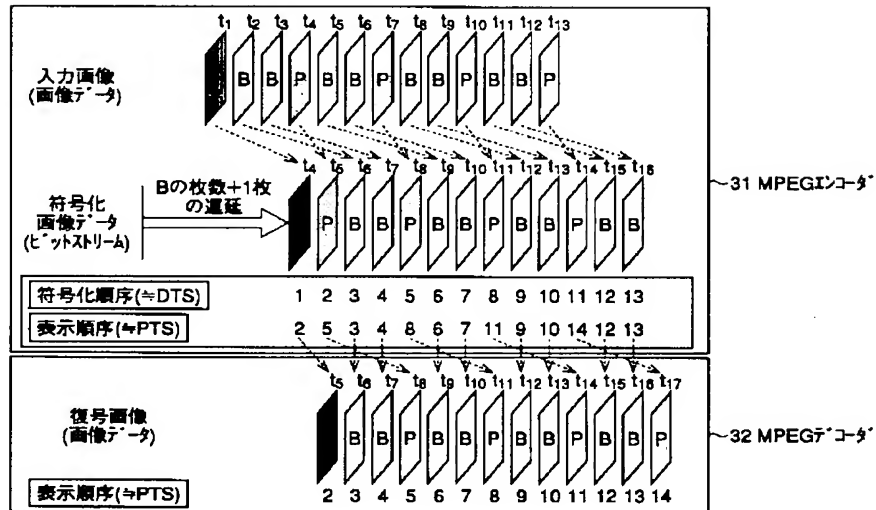


【図8】

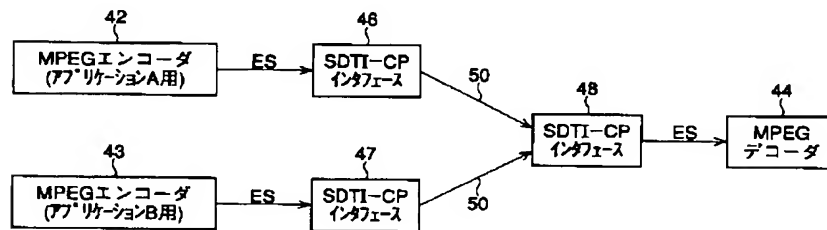


SDTI-CP：放送局内

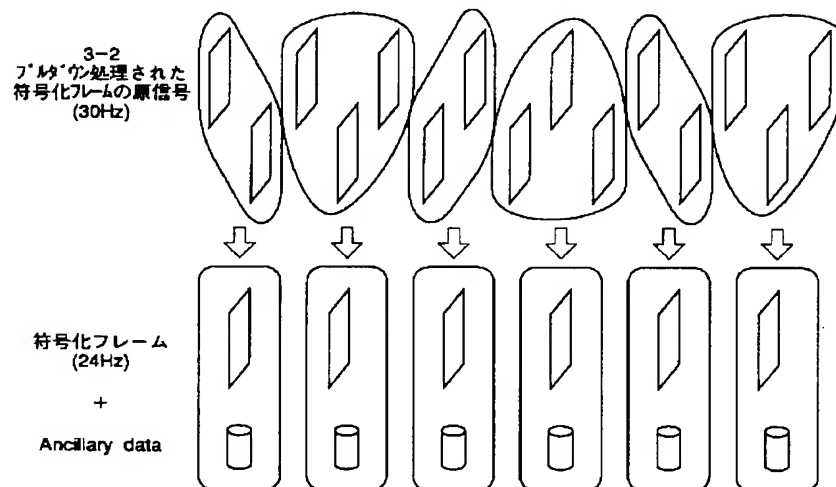
【図6】



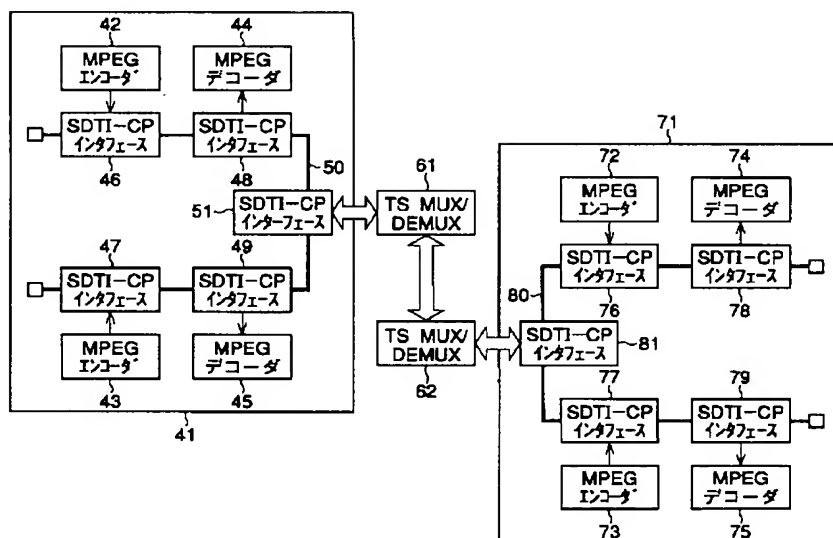
【図9】



【図10】



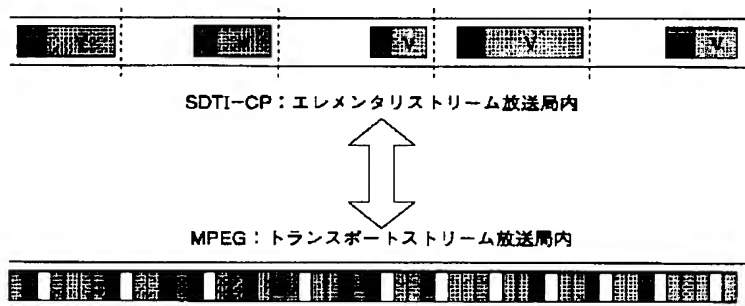
【図11】



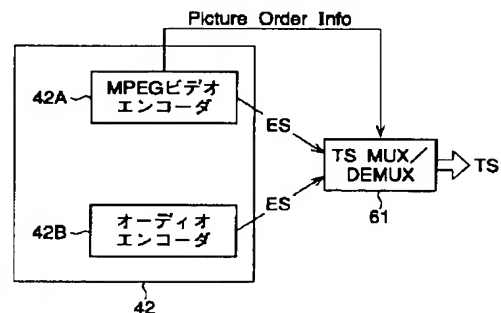
【図28】

| Data_ID | Data_type      |
|---------|----------------|
| 00      | FORBIDDEN      |
| 01      | V-Phase        |
| 02      | H-Phase        |
| 03      | Time code 1    |
| 04      | Time code 2    |
| 05      | Picture Order  |
| 06      | Video Index    |
| 07      | Ancillary data |
| 08      | History data   |
| ...     | ...            |
| 80      | Control flags  |
| ...     | ...            |
| FF      | User data      |

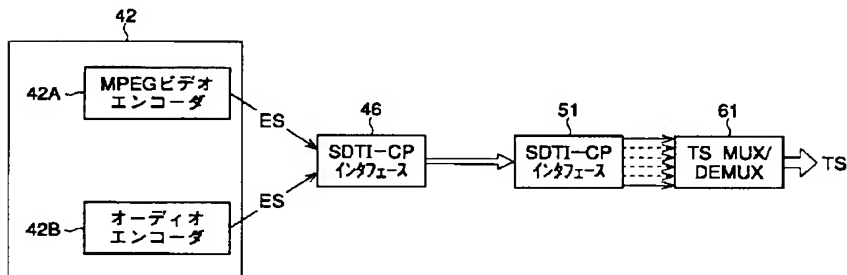
【図12】



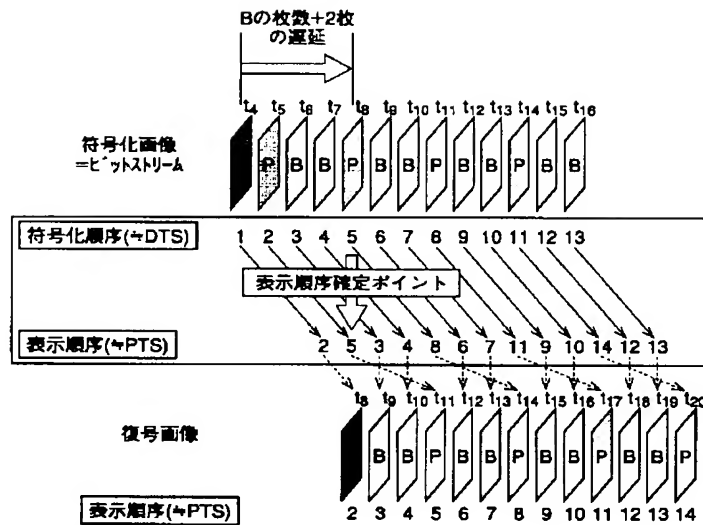
【図22】



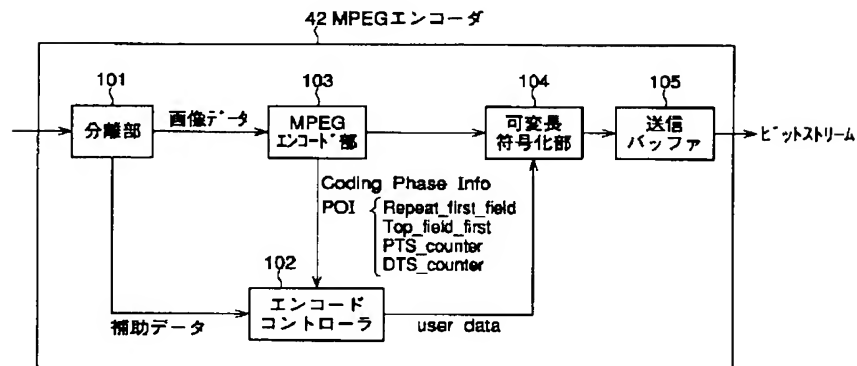
【図13】



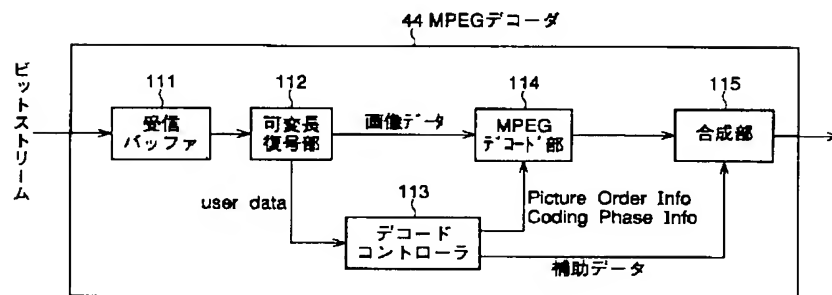
【図14】



【図15】

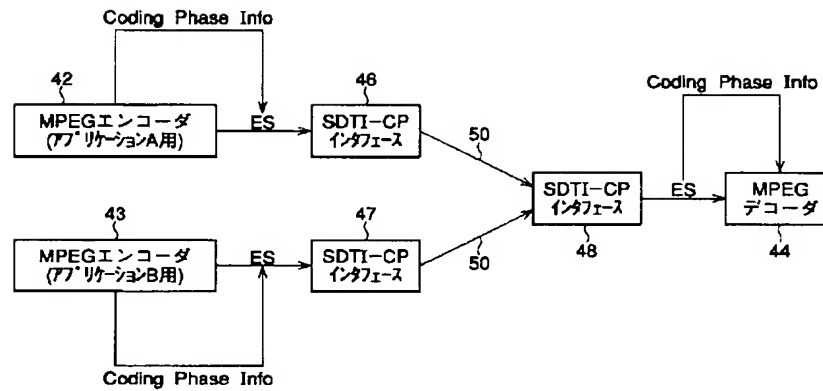


【図16】

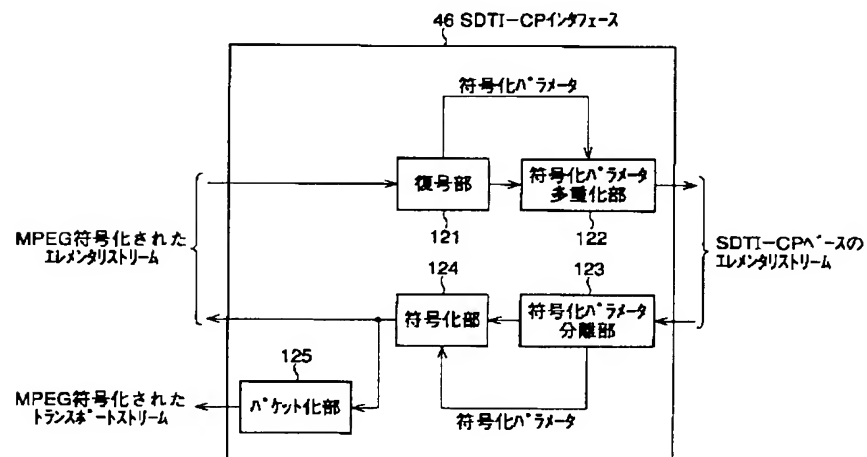




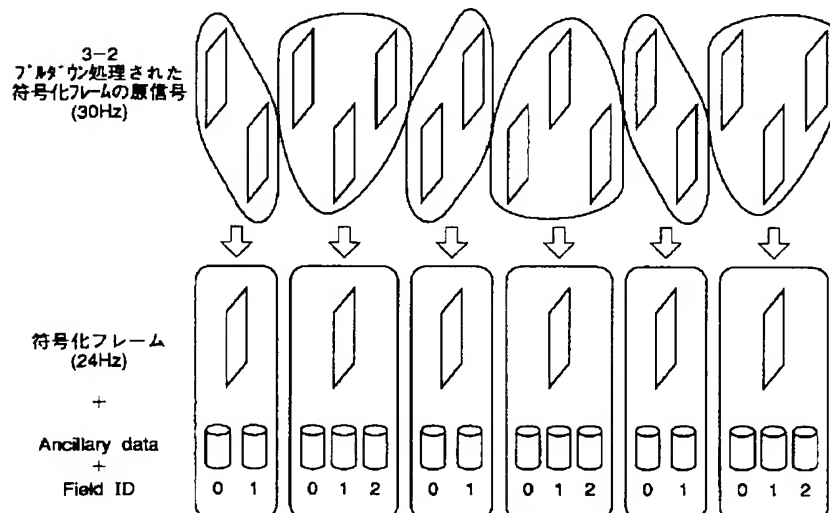
【図17】



【図18】



【図19】



【図20】

|                    |   |   |   |   |   |   |   |   |   |    |    |    |    |
|--------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|
| Frame No           | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Picture type       | I | B | B | P | B | B | P | B | B | I  | B  | B  | P  |
| Repeat_first_field | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 0  | 1  |
| Top_field_first    | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0  | 0  | 1  | 1  |
|                    |   |   |   |   |   |   |   |   |   |    |    |    |    |

(A)

|                    |   |   |   |   |    |    |    |    |    |    |    |    |    |
|--------------------|---|---|---|---|----|----|----|----|----|----|----|----|----|
| Frame No           | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 |
| Picture type       | I | B | B | P | B  | B  | P  | B  | B  | I  | B  | B  | P  |
| Repeat_first_field | 1 | 0 | 1 | 0 | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  |
| Top_field_first    | 1 | 0 | 0 | 1 | 1  | 0  | 0  | 1  | 1  | 0  | 0  | 1  | 1  |
|                    |   |   |   |   |    |    |    |    |    |    |    |    |    |
| PTS_counter        | 0 | 3 | 5 | 8 | 10 | 13 | 15 | 18 | 20 | 23 | 25 | 28 | 30 |

(B)

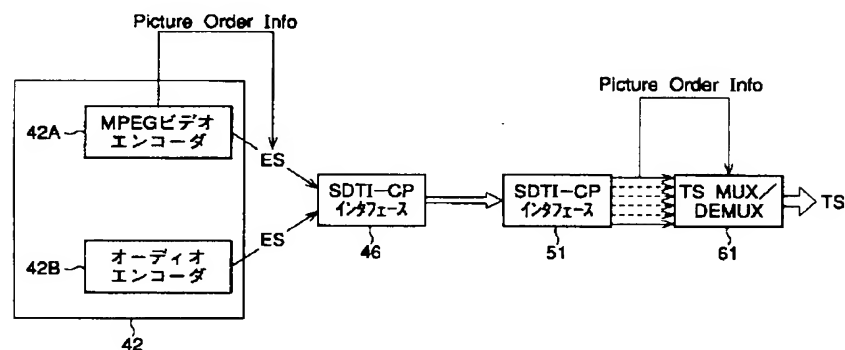
|                    |     |   |   |   |   |    |    |    |    |    |    |    |    |
|--------------------|-----|---|---|---|---|----|----|----|----|----|----|----|----|
| Frame No           | 1   | 4 | 2 | 3 | 7 | 5  | 6  | 10 | 8  | 9  | 13 | 11 | 12 |
| Picture type       | I   | P | B | B | P | B  | B  | I  | B  | B  | P  | B  | B  |
| Repeat_first_field | 1   | 0 | 0 | 1 | 1 | 1  | 0  | 0  | 0  | 1  | 1  | 1  | 0  |
| Top_field_first    | 1   | 1 | 0 | 0 | 0 | 1  | 0  | 0  | 1  | 1  | 1  | 0  | 1  |
|                    |     |   |   |   |   |    |    |    |    |    |    |    |    |
| DTS_counter        | 125 | 0 | 3 | 5 | 8 | 10 | 13 | 15 | 18 | 20 | 23 | 25 | 28 |

(C)

【図23】

| video_sequence()                           | No. of bits | Mnemonic |
|--|-------------|----------|
| next_start_code()                          |             |          |
| sequence_header()                          |             |          |
| sequence_extension()                       |             |          |
| do{  |             |          |
| extension_and_user_data(0)                 |             |          |
| do{  |             |          |
| if(nextbits()--group_start_code){          |             |          |
| group_of_pictures_header()                 |             |          |
| extension_and_user_data(1)                 |             |          |
| }  |             |          |
| picture_header()                           |             |          |
| picture_coding_extension()                 |             |          |
| extension_and_user_data(2)                 |             |          |
| picture_data()                             |             |          |
| }while((nextbits()--picture_start_code) != |             |          |
| (nextbits()--group_start_code))            |             |          |
| if(nextbits() != sequence_end_code){       |             |          |
| sequence_header()                          |             |          |
| sequence_extension()                       |             |          |
| }  |             |          |
| }while(nextbits() != sequence_end_code)    |             |          |
| sequence_end_code                          | 32          | bslbf    |
| }  |             |          |

【図21】



【図26】

| extension_and_user_data(i){           | No. of bits | Mnemonic |
|---------------------------------------|-------------|----------|
| while((i--)>0){                       |             |          |
| if(nextbits()--extension_start_code){ |             |          |
| if(nextbits()--extension_start_code)  |             |          |
| extension_data(i)                     |             |          |
| if(nextbits()--user_data_start_code)  |             |          |
| user_data()                           |             |          |
| }                                     |             |          |
| }                                     |             |          |

【図29】

| Syntax     | Bits | Mnemonic |
|------------|------|----------|
| V-Phase(){ |      |          |
| Data_ID    | 8    | bslbf    |
| V-Phase    | 16   | uimsbf   |
| }          |      |          |

【図24】

| sequence_header{                    | No. of bits | Mnemonic |
|-------------------------------------|-------------|----------|
| sequence_header_code                | 32          | bslbf    |
| horizontal_size_value               | 12          | uimsbf   |
| vertical_size_value                 | 12          | uimsbf   |
| aspect_ratio_information            | 4           | uimsbf   |
| frame_rate_code                     | 4           | uimsbf   |
| bit_rate_value                      | 18          | uimsbf   |
| marker_bit                          | 1           | "1"      |
| vbv_buffer_size_value               | 10          | uimsbf   |
| constrained_parameters_flag         | 1           |          |
| load_intra_quantiser_matrix         | 1           |          |
| If(load_intra_quantiser_matrix)     |             |          |
| intra_quantiser_matrix [64]         | 8*64        | uimsbf   |
| load_non_intra_quantiser_matrix     | 1           |          |
| If(load_non_intra_quantiser_matrix) |             |          |
| non_intra_quantiser_matrix [64]     | 8*64        | uimsbf   |
| next_start_code()                   |             |          |
| }                                   |             |          |

【図27】

| user_data{   | No. of bits | Mnemonic |
|--|-------------|----------|
| user_data_start_code   | 32          |          |
| If(i==0){  |             |          |
| while(nextbits()!="0000 0000 0000 0000 0001" ){              |             |          |
| If(nextbits()=="V-phase" )                                   |             |          |
| V-phase()  |             |          |
| Else If(nextbits()=="H-phase" )                              |             |          |
| H-phase()  |             |          |
| {  |             |          |
| Else If(i==2){   |             |          |
| while(nextbits()!="0000 0000 0000 0000 0001" ){              |             |          |
| If(nextbits()=="Time code 1" )! (nextbits()=="Time code 2" ) |             |          |
| Time_code()  |             |          |
| Else If(nextbits()=="Picture Order" )                        |             |          |
| Picture_order()  |             |          |
| {  |             |          |
| while(nextbits()!="0000 0000 0000 0000 0001" ){              |             |          |
| If(nextbits()=="Ancillary_data" )                            |             |          |
| Ancillary_data()   |             |          |
| Else If(nextbits()=="History data" )                         |             |          |
| History_data()   |             |          |
| {  |             |          |
| If(nextbits()=="User data" )                                 |             |          |
| user_data  |             |          |
| }  |             |          |

【図31】

| Syntax                  | Bits | Mnemonic |
|-------------------------|------|----------|
| Picture_order{          |      |          |
| Data_ID                 | 8    | bslbf    |
| DTS_presence            | 1    | bslbf    |
| PTS_counter             | 7    | uimsbf   |
| If(DTS_presence=="1" ){ |      |          |
| Marker_bits             | 1    | bslbf    |
| DTS_counter             | 7    | uimsbf   |
| }                       |      |          |
| }                       |      |          |

【図25】

| sequence_extension{             | No. of bits | Mnemonic |
|---------------------------------|-------------|----------|
| extension_start_code            | 32          | bslbf    |
| extension_start_code_identifier | 4           | uimsbf   |
| profile_and_level_indication    | 8           | uimsbf   |
| progressive_sequence            | 1           | uimsbf   |
| chroma_format                   | 2           | uimsbf   |
| horizontal_size_extension       | 2           | uimsbf   |
| vertical_size_extension         | 2           | uimsbf   |
| bit_rate_extension              | 12          | uimsbf   |
| marker_bit                      | 1           | bslbf    |
| vbv_buffer_size_extension       | 8           | uimsbf   |
| low_delay                       | 1           | uimsbf   |
| frame_rate_extension_n          | 2           | uimsbf   |
| frame_rate_extension_d          | 5           | uimsbf   |
| next_start_code()               |             |          |
| }                               |             |          |

【図30】

| Syntax   | Bits | Mnemonic |
|----------|------|----------|
| H-Phase{ |      |          |
| Data_ID  | 8    | bslbf    |
| H-Phase  | 8    | uimsbf   |
| }        |      |          |

【図32】

| Syntax                                  | Bits | Mnemonic |
|---|------|----------|
| Ancillary_data{                         |      |          |
| Data_ID                                 | 8    | bslbf    |
| Field_ID                                | 2    | bslbf    |
| Line_number                             | 14   | uimsbf   |
| Ancillary_data_length                   | 16   | uimsbf   |
| Marker_bits                             | 1    | bslbf    |
| For(j=0; j<Ancillary_data_length; j++){ |      |          |
| Ancillary_data_payload                  | 22   | uimsbf   |
| Marker_bits                             | 1    | bslbf    |
| }                                       |      |          |
| While(!bytealigned())                   |      |          |
| Zero_bit                                | 1    | "0"      |
| }                                       |      |          |

【図33】

| group_of_picture_header{ | No. of bits | Mnemonic |
|--------------------------|-------------|----------|
| group_start_code         | 32          | bslbf    |
| time_code                | 25          | bslbf    |
| closed_gop               | 1           | uimsbf   |
| broken_link              | 1           | uimsbf   |
| next_start_code()        |             |          |
| }                        |             |          |

【図34】

| picture_header()                                      | No. of bits | Mnemonic |
|---|-------------|----------|
| picture_start_code                                    | 32          | bslbf    |
| temporal_reference                                    | 10          | uimsbf   |
| picture_coding_type                                   | 3           | uimsbf   |
| vbv_delay   | 18          | uimsbf   |
| if(picture_coding_type==2    picture_coding_type==3){ |             |          |
| full_pel_forward_vector                               | 1           |          |
| forward_f_code  | 3           | uimsbf   |
| }   |             |          |
| if(picture_coding_type==3){                           |             |          |
| full_pel_backward_vector                              | 1           |          |
| backward_f_code                                       | 3           | uimsbf   |
| }   |             |          |
| while(nextbits()!="1"){                               |             |          |
| extra_bit_picture /*with the value "1" */             | 1           | uimsbf   |
| extra_information_picture                             | 8           |          |
| }   |             |          |
| extra_bit_picture /*with the value "0" */             | 1           | uimsbf   |
| next_start_code()                                     |             |          |
| }   |             |          |

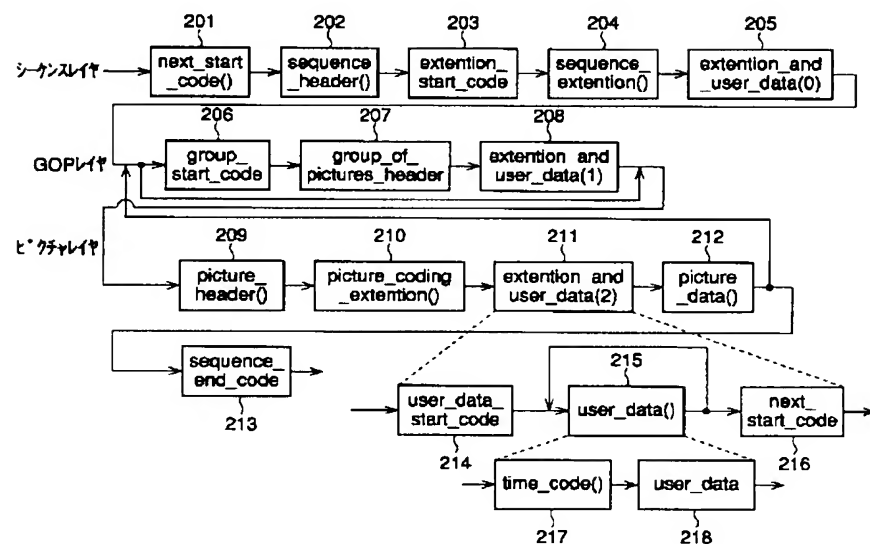
【図36】

| picture_data()                       | No. of bits | Mnemonic |
|--------------------------------------|-------------|----------|
| do{                                  |             |          |
| slice()                              |             |          |
| }while(nextbits()--slice_start_code) |             |          |
| next_start_code()                    |             |          |
| }                                    |             |          |

【図35】

| picture_coding_extension()            | No. of bits | Mnemonic |
|---------------------------------------|-------------|----------|
| extension_start_code                  | 32          | bslbf    |
| extension_start_code_identifier       | 4           | uimsbf   |
| f_code[0][0] /*forward horizontal */  | 4           | uimsbf   |
| f_code[0][1] /*forward vertical */    | 4           | uimsbf   |
| f_code[1][0] /*backward horizontal */ | 4           | uimsbf   |
| f_code[1][1] /*backward vertical */   | 4           | uimsbf   |
| intra_dc_precision                    | 2           | uimsbf   |
| picture_structure                     | 2           | uimsbf   |
| top_field_first                       | 1           | uimsbf   |
| frame_pred_frame_dct                  | 1           | uimsbf   |
| concealment_motion_vectors            | 1           | uimsbf   |
| q_scale_type                          | 1           | uimsbf   |
| intra_vic_format                      | 1           | uimsbf   |
| alternate_scan                        | 1           | uimsbf   |
| repeat_first_field                    | 1           | uimsbf   |
| chroma_420_type                       | 1           | uimsbf   |
| progressive_frame                     | 1           | uimsbf   |
| composite_display_flag                | 1           | uimsbf   |
| if(composite_display_flag){           |             |          |
| v_axis                                | 1           | uimsbf   |
| field_sequence                        | 3           | uimsbf   |
| sub_carrier                           | 1           | uimsbf   |
| burst_amplitude                       | 7           | uimsbf   |
| sub_carrier_phase                     | 8           | uimsbf   |
| }                                     |             |          |
| next_start_code()                     |             |          |
| }                                     |             |          |

【図37】



フロントページの続き

Fターム(参考) 5C059 KK11 LA09 MA00 ME01 PP05  
PP06 PP07 RB02 RB10 RB11  
RC04 RC26 SS06 SS08 SS11  
UA02 UA05 UA32